University of Alberta

Library Release Form

Name of Author: David Michael White

Title of Thesis: Robust Controller Synthesis Utilizing Kharitonov's Theorem

Degree: Master of Science

Year this Degree Granted: 1999

University of Alberta

Library Release Form

Name of Author: [illegible]

Title of Thesis: [illegible]

Degree: Master of Science

Year this Degree Granted: [illegible]

"So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality."
— *Albert Einstein*

University of Alberta

Robust Controller Synthesis Utilizing Kharitonov's Theorem

by

David Michael White ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

in

Control Systems

Department of Electrical Engineering

Edmonton, Alberta
Fall 1999

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Robust Controller Synthesis Utilizing Kharitonov's Theorem** submitted by David Michael White in partial fulfillment of the requirements for the degree of **Master of Science** in *Control Systems.*

Abstract

Significant use is being made of Kharitonov's Theorem in studying the robustness of dynamic systems. However, it is still perceived as a difficult theorem to apply in a robust controller synthesis context. This thesis examines the use of simultaneous stabilization techniques to utilize Kharitonov's Theorem in designing MIMO state and output feedback controllers, using Syncrude Canada Ltd.'s steam letdown system as a case study.

Acknowledgements

I would like to thank Dr. Ray Rink for his guidance and for his considerable contributions to the *SynSim* dynamic model. I would also like to thank him for being the apotheosis of patience.

Also thanks to Dr. Horacio Marquez for taking on the role of my thesis supervisor at the last minute.

Dr. Wosang Young kindly allowed me to use a suite of MATLAB functions she created for use with Kharitonov's Theorem. They have been very useful and for that I am appreciative.

Thanks also to Kent Gooden for initiating the collaborative research and development grant which led into all this work. I would also like to thank him for providing numerous insights into how the integrated system works.

I am grateful to Peter Morgan for his mentorship and suggesting the letdown system as a suitable case study for this work.

Finally, I would like to thank my parents, Dean and Sharon, and sister Dena for always being there for me.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

This research was undertaken as part of a larger project for Syncrude Canada Ltd. to develop a high order nonlinear dynamic model of their utility steam system and associated controls. There is a rising demand for robustly stable process operations while increasing automation [18]. This necessitates steam system controls that provide stable responses to operational disturbances. Design deficiencies, in either the actual processes or the control systems, can be costly and even catastrophic if they are discovered only after an incident. Dynamic simulation analysis is a useful tool to avoid such costly revelations. *SynSim* is a desktop computer based simulation developed jointly by members of the University of Alberta Departments of Electrical and Chemical Enginering, and members of the Syncrude Canada Ltd. Utilities Technical Team. The model simulates all of the utility boilers, CO boilers, once-through steam generators, backpressure turbine generators, condensing turbine generator, gas turbine generators, all of the major steam headers, as well as numerous other pieces of equipment. This requires over four hundred state variables and thousands of parameters. The purpose of the *SynSim* model is to investigate the response of Syncrude's utility system, especially the steam system, under plant upset conditions such as tieline separations, boiler trips, steam turbine generator trips, and numerous other possible events. The *SynSim* model has been used in testing system stability limits,

future plant expansion configurations, robust control algorithms, and was used as a platform for this work.

1.2 Robust Control

The primary feature of robust control is the explicit consideration of uncertainties, which usually result from unmodeled system dynamics, external disturbances, approximations due to linearization, and measurement errors. Nearly all classical control theory is based on assumptions about the plant or process, and utilizes a nominal mathematical model which is assumed to be linear and time invariant. For example, in Ziegler-Nichols tuning of PID controller parameters, the model used in deriving the equations is simply a first-order system plus a transportation lag [17]. If the actual plant is higher order, has variable transportation lag, or has other unmodeled nonlinearities, the designer must be very conservative when implementing a control system with such methods, and even then it is probably necessary to thoroughly test the proposed control system design to ensure its stability or performance. Some uncertainties can be removed or minimized by using a higher-order model or by improving measurement, but it is inevitable that some uncertainties will remain. This is because all models are inaccurate to some extent. Plant loads and operating points change, disturbances and noise are always present, and even plant configurations do not remain constant. It is therefore desirable to be able to guarantee the stability or some other performance criteria for the system in the presence of uncertainty. Using robust control theory, we can assign uncertainty models to the system and be less conservative in a control system design while still guaranteeing the same design requirements. The same theories can be used to analyze an existing system to more accurately determine the stability margins.

There are two main types of uncertainties [29]. Parametric uncertainties are those that can be compensated for by correcting the model parameter values. Structural uncertainties are brought about by an imperfect structure of the model.

Parametric uncertainties with an independent uncertainty structure are represented by interval polynomials, which have the general form:

$$p(s,q) = [q_0^-, q_0^+] + [q_1^-, q_1^+]s + [q_2^-, q_2^+]s^2 + \cdots + [q_n^-, q_n^+]s^n \tag{1.1}$$

where $q_i^- \leq q_i^+$ represent lower and upper bounds for the continuous set q_i. The set $\{q_i, i = 1, \cdots, n\}$ is represented as $q \in Q$, where Q is the uncertainty bounding set: $Q = \{q \in \Re^n : q_i^- \leq q_i \leq q_i^+,\ i = 1, \cdots, n\}$.

Structural uncertainties are handled by either additive or multiplicative perturbation sets [37]. A system model described by a set of additive perturbations is represented by:

$$\mathbf{\Pi} = \mathbf{P} + \mathbf{W}_1 \mathbf{\Delta} \mathbf{W}_2 \tag{1.2}$$

where $\mathbf{\Pi}$ is the set of uncertain plants, $\mathbf{P} \in \mathbf{\Pi}$ is the nominal plant, $\mathbf{W}_1$ and $\mathbf{W}_2$ are weighting functions, and $\mathbf{\Delta}$ is the modeling error. Similarly, a system model described by a set of multiplicative perturbations is shown as:

$$\mathbf{\Pi} = (\mathbf{I} + \mathbf{W}_1 \mathbf{\Delta} \mathbf{W}_2)\mathbf{P} \tag{1.3}$$

where $\mathbf{I}$ is the identity matrix. However, we will only be concerned with the parametric approach here.

A good definition of robustness is as follows [2]:

> Given a family $\mathcal{F}$ and some property $\mathbf{P}$, if every member $f \in \mathcal{F}$ has property $\mathbf{P}$, then the family $\mathcal{F}$ is robust with respect to property $\mathbf{P}$.

Now, robust $\mathcal{D}$-stability means the following [3]:

> Let $\mathcal{D} \subseteq \mathcal{C}$ ($\mathcal{C}$ is the complex plane), then the polynomial $p(s)$ is $\mathcal{D}$-stable if all its roots lie in the region $\mathcal{D}$. A family of interval polynomials $\mathcal{P} = \{p(\cdot, q) : q \in Q\}$ is robustly $\mathcal{D}$-stable if, $\forall q \in Q$, all roots of $p(s,q)$ lie in $\mathcal{D}$. There are two special cases of robust $\mathcal{D}$-stability: in the case where $\mathcal{D}$ is the open left half plane $\mathcal{P}$ is said to be robustly stable and in the case when $\mathcal{D}$ is the open unit disc $\mathcal{P}$ is said to be robustly Schur stable.

1.3 Case Study

Syncrude's Mildred Lake plant consists of a tarsand mine, a bitumen extraction process, and an upgrader to produce a high quality synthetic crude oil. Roughly 14% of Canada's domestic crude oil requirement is currently supplied by this plant, and there are plans to greatly increase this production in the near future. The Utilities Department is critical to plant operation because it supplies steam, water, electricity, air, and nitrogen to the various processes on the site. The main components of the steam system are three fuel gas fired utility boilers and two CO boilers which, with supplemental fuel gas firing, burn carbon monoxide and other waste gases from the cokers. These boilers supply a 900# ring header. The steam is then distributed to 600#, 150#, and 50# headers. There are four steam turbine generators and two gas turbine generators. Three of the steam turbine generators take their inlet steam from the 900# header and exhaust into the 50# header, thereby controlling its pressure, and are referred to as backpressure turbines. The fourth steam turbine generator is a condensing turbine. A number of letdowns also serve to regulate the various header pressures. Because of the steam production from waste gases and the power generation as part of a steam letdown, it is an integrated energy facility, or cogeneration plant, and not a typical utility. Here, the primary focus is on reliable steam production and electrical power is a by-product and the controls are tuned for robustness more than optimization.

The header system causes numerous system stability problems. Traditionally, steam turbine generators are connected directly to a boiler and there is no steam header system. At Syncrude's utility plant, the boilers feed and the steam turbines draw from the same 900# header, and the backpressure units exhaust into the same 50# header, resulting in a strongly coupled or affine system. This lack of component autonomy makes the utility plant acutely sensitive to upsets in the steam system. Figures 1.1 and 1.2 demonstrate the 900/600# and the 600/50# headers interacting

in an undesirable manner. These plots were generated using *SynSim*, but the phenomenon has also been observed in actual plant data. Because the letdown system is difficult to control due to its affine nature, it was selected as the case study for this work.

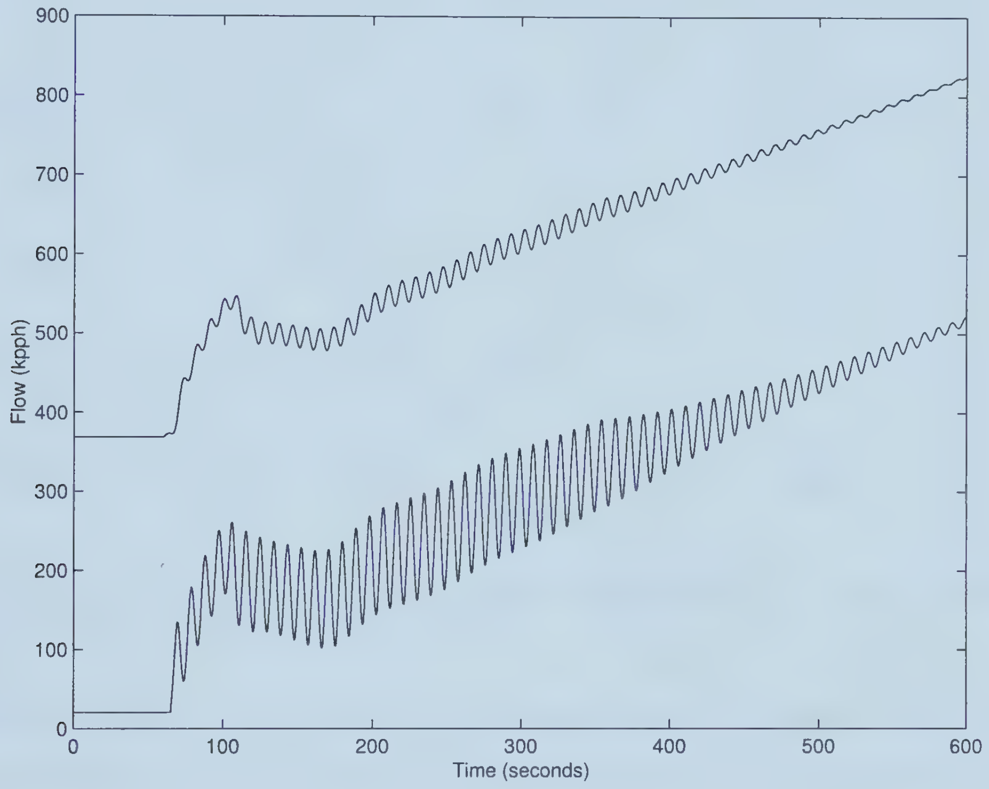

Figure 1.1: 900/600# flow (top) and 600/50# flow (bottom) interactions for a back-pressure turbine generator trip.

1.4 Thesis Outline

The objective of this thesis is to advocate Kharitonov's Theorem as a useful tool for designing or tuning controllers, and to demonstrate several methods of robust controller synthesis by applying Kharitonov's Theorem.

The purpose of Chapter 2 is to provide an overview of Kharitonov's Theorem and several related theorems and tools which are particularly useful. The most important section introduces a simultaneous stabilization algorithm which can simultaneously stabilize and optimize a number of plants with a single MIMO controller.

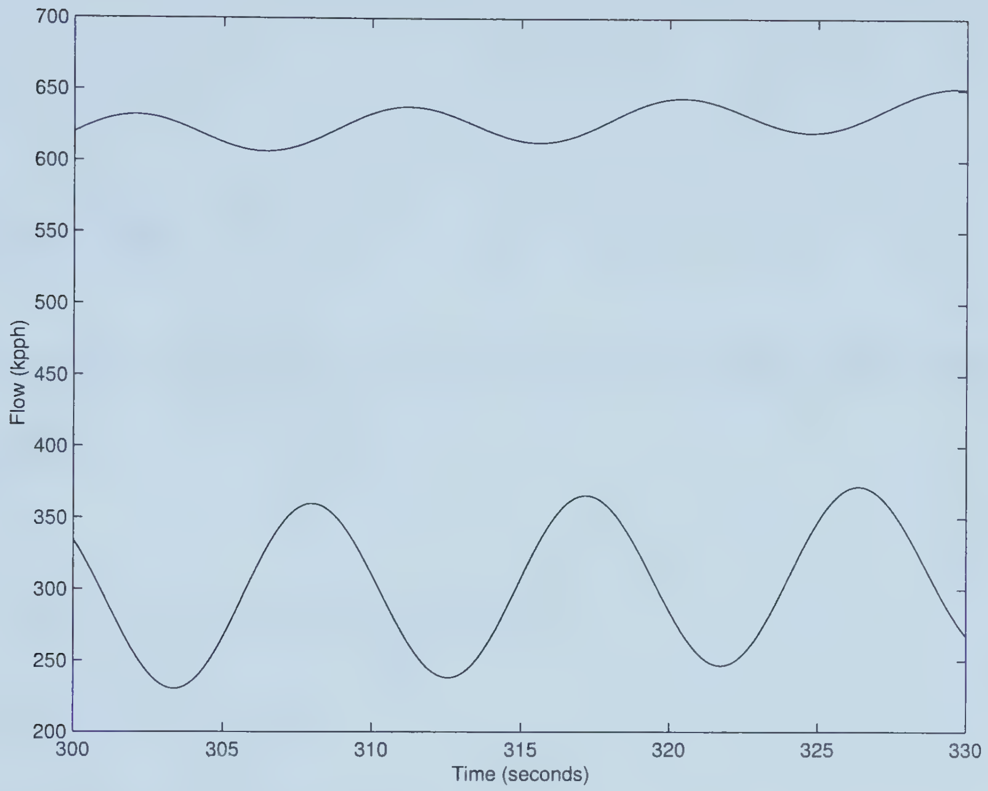

Figure 1.2: Close up of 900/600# flow (top) and 600/50# flow (bottom) interactions. Note that they are nearly 180° out of phase.

Chapter 3 goes over the modeling processes required. A simple first principles model is developed for the design of the controllers, and a procedure for obtaining system identification models is presented.

In Chapter 4, two MIMO controllers are designed such that some of the model uncertainty is accounted for. The first is a state feedback controller and the second is an output feedback controller. The performance of both controllers is discussed and compared to that of the existing cascaded PI controllers.

Chapter 2

Kharitonov's Theorem and Related Tools

2.1 Kharitonov's Theorem

Kharitonov's Theorem [4] [24]:

Given the interval polynomial:

$$p(s,q) = q_0 + q_1 s + q_2 s^2 + \cdots + q_n s^n \tag{2.1}$$

$$p(s,q) = [q_0^-, q_0^+] + [q_1^-, q_1^+]s + [q_2^-, q_2^+]s^2 + \cdots + [q_n^-, q_n^+]s^n \tag{2.2}$$

Where $q_i^- \leq q_i^+$ represent lower and upper bounds for the continuous set q_i. An polynomial family $\mathcal{P}$ with invariant degree is robustly stable if and only if the following four Kharitonov polynomials are strictly stable:

$$K_1(s) = q_0^- + q_1^- s + q_2^+ s^2 + q_3^+ s^3 + \cdots \tag{2.3}$$

$$K_2(s) = q_0^+ + q_1^+ s + q_2^- s^2 + q_3^- s^3 + \cdots \tag{2.4}$$

$$K_3(s) = q_0^+ + q_1^- s + q_2^- s^2 + q_3^+ s^3 + \cdots \tag{2.5}$$

$$K_4(s) = q_0^- + q_1^+ s + q_2^+ s^2 + q_3^- s^3 + \cdots \tag{2.6}$$

Note that the assumption of invariant degree implies that $0 \notin q_n$, in other words, q_n^- and q_n^+ must have the same sign. The four Kharitonov polynomials have a very

simple derivation. Separate $p(j\omega, q)$, into its real and imaginary components:

$$p(j\omega, q) = q_0 + q_1 j\omega - q_2\omega^2 - q_3 j\omega^3 + q_4\omega^4 + q_5 j\omega^5 - q_6\omega^6 - \cdots \quad (2.7)$$

$$\Re p(j\omega, q) = q_0 - q_2\omega^2 + q_4\omega^4 - q_6\omega^6 + q_8\omega^8 - q_{10}\omega^{10} + \cdots \quad (2.8)$$

$$\Im p(j\omega, q) = q_1 - q_3\omega^3 + q_5\omega^5 - q_7\omega^7 + q_9\omega^9 - q_{11}\omega^{11} + \cdots \quad (2.9)$$

Notice that the real and imaginary components are decoupled with respect to q. An interval polynomial (or interval polynomial family),

$$\mathcal{P} = \{p(s, q) : q \in Q\} \quad (2.10)$$

by definition has an independent uncertainty structure. In other words, each coefficient depends continuously on q and Q is a box. Taking the minimum and maximum values at each frequency for the real and imaginary parts of $p(j\omega, q)$ gives us:

$$\min \Re p(j\omega, q) = q_0^- - q_2^+\omega^2 + q_4^-\omega^4 - q_6^+\omega^6 + q_8^-\omega^8 - \cdots \quad (2.11)$$

$$\max \Re p(j\omega, q) = q_0^+ - q_2^-\omega^2 + q_4^+\omega^4 - q_6^-\omega^6 + q_8^+\omega^8 - \cdots \quad (2.12)$$

$$\min \Im p(j\omega, q) = q_1^- - q_3^+\omega^3 + q_5^-\omega^5 - q_7^+\omega^7 + q_9^-\omega^9 - \cdots \quad \omega \geq 0 \quad (2.13)$$

$$\min \Im p(j\omega, q) = q_1^+ - q_3^+\omega^3 + q_5^+\omega^5 - q_7^+\omega^7 + q_9^+\omega^9 - \cdots \quad \omega < 0 \quad (2.14)$$

$$\max \Im p(j\omega, q) = q_1^+ - q_3^+\omega^3 + q_5^+\omega^5 - q_7^+\omega^7 + q_9^+\omega^9 - \cdots \quad \omega \geq 0 \quad (2.15)$$

$$\max \Im p(j\omega, q) = q_1^- - q_3^+\omega^3 + q_5^-\omega^5 - q_7^+\omega^7 + q_9^-\omega^9 - \cdots \quad \omega < 0 \quad (2.16)$$

A Kharitonov rectangle, which represents all of the possible values of $p(j\omega, q)$ if $p(s, q)$ has an independent uncertainty structure, is formed by using these minima and maxima as its vertices and each vertex corresponds to a Kharitonov polynomial, as shown in Fig. 2.1. $K_1(s)$ corresponds to the minimum real and minimum imaginary components. $K_2(s)$ corresponds to the maximum real and maximum imaginary components. If $\omega \geq 0$, then $K_3(s)$ corresponds to the maximum real and minimum imaginary components and $K_4(s)$ corresponds to the minimum real and maximum imaginary components. Otherwise, if $\omega < 0$, then $K_3(s)$ corresponds to the minimum real and

maximum imaginary components and $K_4(s)$ corresponds to the maximum real and minimum imaginary components. This graphical representation of the Kharitonov polynomials makes it easy to see why they are sometimes referred to as vertex polynomials.

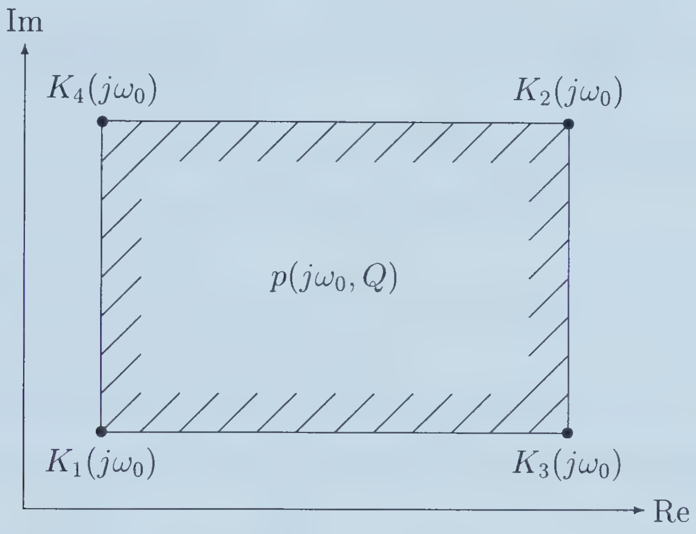

Figure 2.1: A Kharitonov rectangle evaluated at $s = j\omega_0$.

The polynomial:

$$p(s,q) = \sum_{i=0}^{n} a_i(q)s^i \tag{2.17}$$

has an independent uncertainty structure if and only if each component q_i of q enters into only one coefficient a_i. Moreover, an interval polynomial family can be lumped into a second equivalent interval polynomial family where each polynomial $p(s,q)$ is of the form:

$$p(s,q) = \sum_{i=0}^{n} q_i s^i \tag{2.18}$$

and each family is completely described by:

$$p(s,q) = \sum_{i=0}^{n} [q_i^-, q_i^+] s^i \tag{2.19}$$

where $[q_i^-, q_i^+]$ represents the bounding interval for the i^{th} component of uncertainty q_i.

After manipulating a dynamic system into a polynomial problem with coefficients depending on uncertain parameters. the issue of uncertainty structure arises. Kharitonov's Theorem is an idealized one where each uncertain parameter enters into only one coefficient (*ie.* an independent uncertainty structure).

2.1.1 Matrix Formulation of Kharitonov's Theorem

Kharitonov's Theorem can be reconstructed in terms of a MIMO state-space realization [9]. Consider the set of uncertain companion matrices:

$$\mathcal{A} \triangleq \left\{ \begin{bmatrix} \mathbf{0}_{(n-1)\times 1} & & \mathbf{I}_{n-1} \\ -\beta_0 & \cdots & -\beta_{n-1} \end{bmatrix}, \beta_i^- < \beta_i < \beta_i^+, i = 0, \cdots, n-1 \right\} \tag{2.20}$$

where $\beta^- \leq \beta^-$ are given uncertainty bounds. Then every matrix in the family of matrices $\mathcal{A}$ is stable if and only if the following four Kharitonov matrices are stable [9]:

$$\mathbf{A_1} \triangleq \begin{bmatrix} & \mathbf{0}_{(n-1)\times 1} & & \mathbf{I}_{n-1} & \\ \cdots & -\beta_{n-4}^- & -\beta_{n-3}^- & -\beta_{n-2}^+ & -\beta_{n-1}^+ \end{bmatrix} \tag{2.21}$$

$$\mathbf{A_2} \triangleq \begin{bmatrix} & \mathbf{0}_{(n-1)\times 1} & & \mathbf{I}_{n-1} & \\ \cdots & -\beta_{n-4}^+ & -\beta_{n-3}^- & -\beta_{n-2}^- & -\beta_{n-1}^+ \end{bmatrix} \tag{2.22}$$

$$\mathbf{A_3} \triangleq \begin{bmatrix} & \mathbf{0}_{(n-1)\times 1} & & \mathbf{I}_{n-1} & \\ \cdots & -\beta_{n-4}^+ & -\beta_{n-3}^+ & -\beta_{n-2}^- & -\beta_{n-1}^- \end{bmatrix} \tag{2.23}$$

$$\mathbf{A_4} \triangleq \begin{bmatrix} & \mathbf{0}_{(n-1)\times 1} & & \mathbf{I}_{n-1} & \\ \cdots & -\beta_{n-4}^- & -\beta_{n-3}^+ & -\beta_{n-2}^+ & -\beta_{n-1}^- \end{bmatrix} \tag{2.24}$$

The *Polynomial Toolbox* [19] by Kwakernaak *et al* for MATLAB is of particular use when generating or manipulating polynomial matrices. and their `blcomp` command may be used to generate the above block companion matrices.

2.2 Robustness Analysis

2.2.1 The Zero Exclusion Condition

The Kharitonov rectangle at frequency ω_0 (see Fig. 2.1) is described by $p(j\omega_0, Q)$. The four vertices of the rectangle are obtained by evaluating the four Kharitonov

polynomials at $s = j\omega_0$. Numerous Kharitonov rectangles can be plotted using a frequency sweep over a range of ω values.

The Zero Exclusion Condition is a theorem which states that an interval polynomial family with invariant degree and at least one stable member is robustly stable if and only if $z = 0$ is excluded from the Kharitonov rectangle at all non-negative frequencies, in other words:

$$0 \notin p(j\omega, Q), \quad \forall\, \omega \geq 0 \tag{2.25}$$

The Kharitonov rectangles must move counterclockwise about the origin in order to have the monotonic phase increase property of Hurwitz polynomials. If the origin is included in one of the rectangles, then it is trivial to see that the phase angle of one of the rectangle corners on the side that passes through the origin will *decrease* with increasing ω, and there is a contradiction of the monotonic phase increase property, at least one of the members of $p(s, Q)$ is not Hurwitz and the family $p(s, Q)$ is not robustly stable.

It is also possible to compute a cutoff frequency, $\omega_c > 0$, such that $0 \notin p(j\omega, Q)$ for all $\omega > \omega_c$. In other words, ω_c is the highest frequency for which it is necessary to test for zero exclusion. Barmish suggests [4] that a conservative and simple method for computing an appropriate cutoff frequency is equating ω_c to the largest real root of the polynomial:

$$f(\omega) = q_n^- \omega^n - \sum_{i=0}^{n-1} q_i^+ \omega^i \tag{2.26}$$

Appendix A.2 presents `cutoff.m`, a MATLAB function for computing the cutoff frequency in this manner.

This theorem offers an interesting graphical approach to robustness analysis: simply plot the Kharitonov rectangles for enough points from $\omega = 0$ to $\omega = \omega_c$ and check for zero exclusion, as shown in Fig. 2.2 and Fig. 2.4. Both plots were generated using the `krsweep.m` MATLAB function, shown in Appendix A.3.

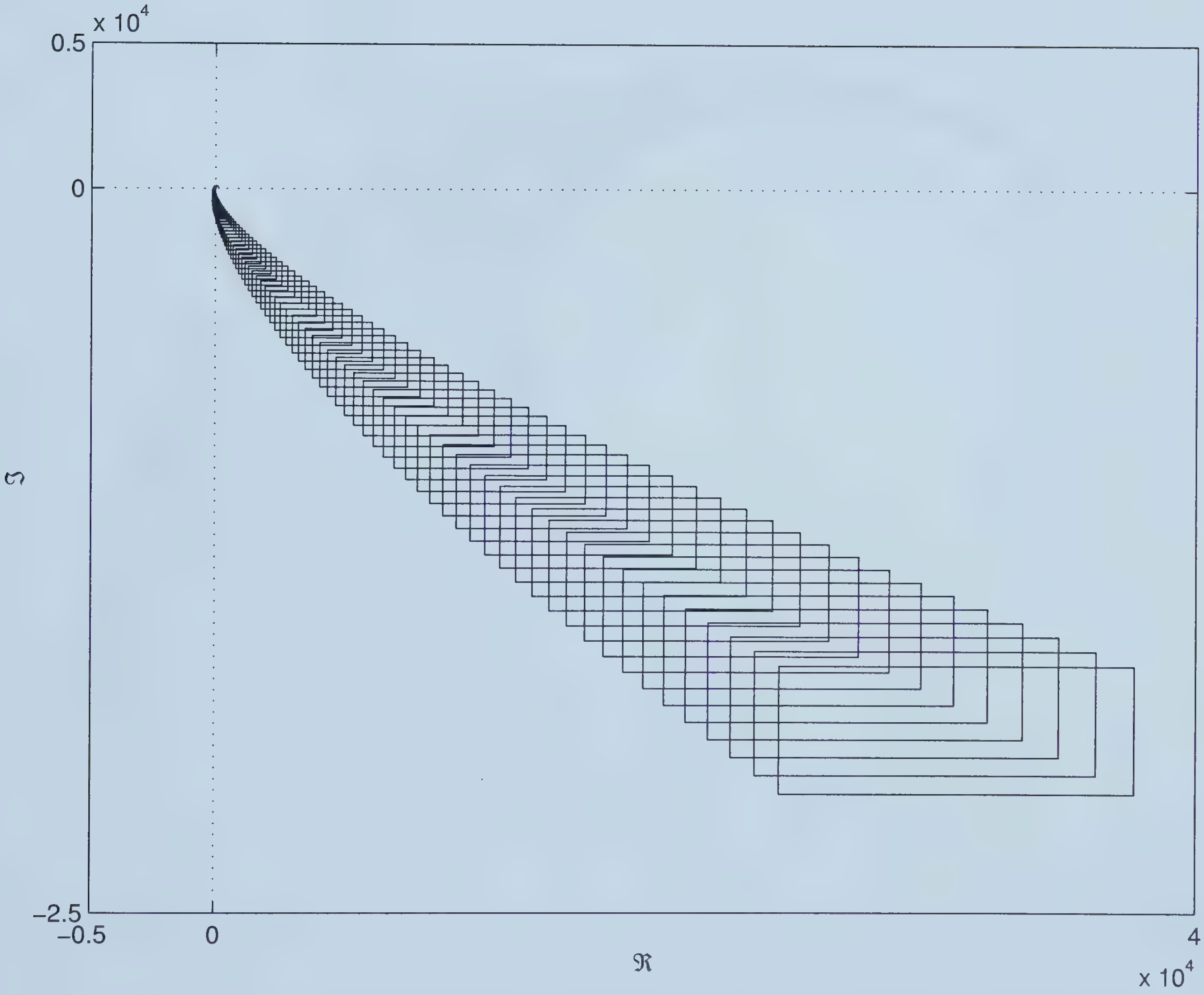

Figure 2.2: The motion of the Kharitonov rectangle for a robustly stable interval polynomial: $p(s) = [110, 115] + [70, 75]s + [30, 32]s^2 + [8, 10]s^3 + [1, 1.5]s^4$, for $\omega = 0$ to $\omega = \omega_c$.

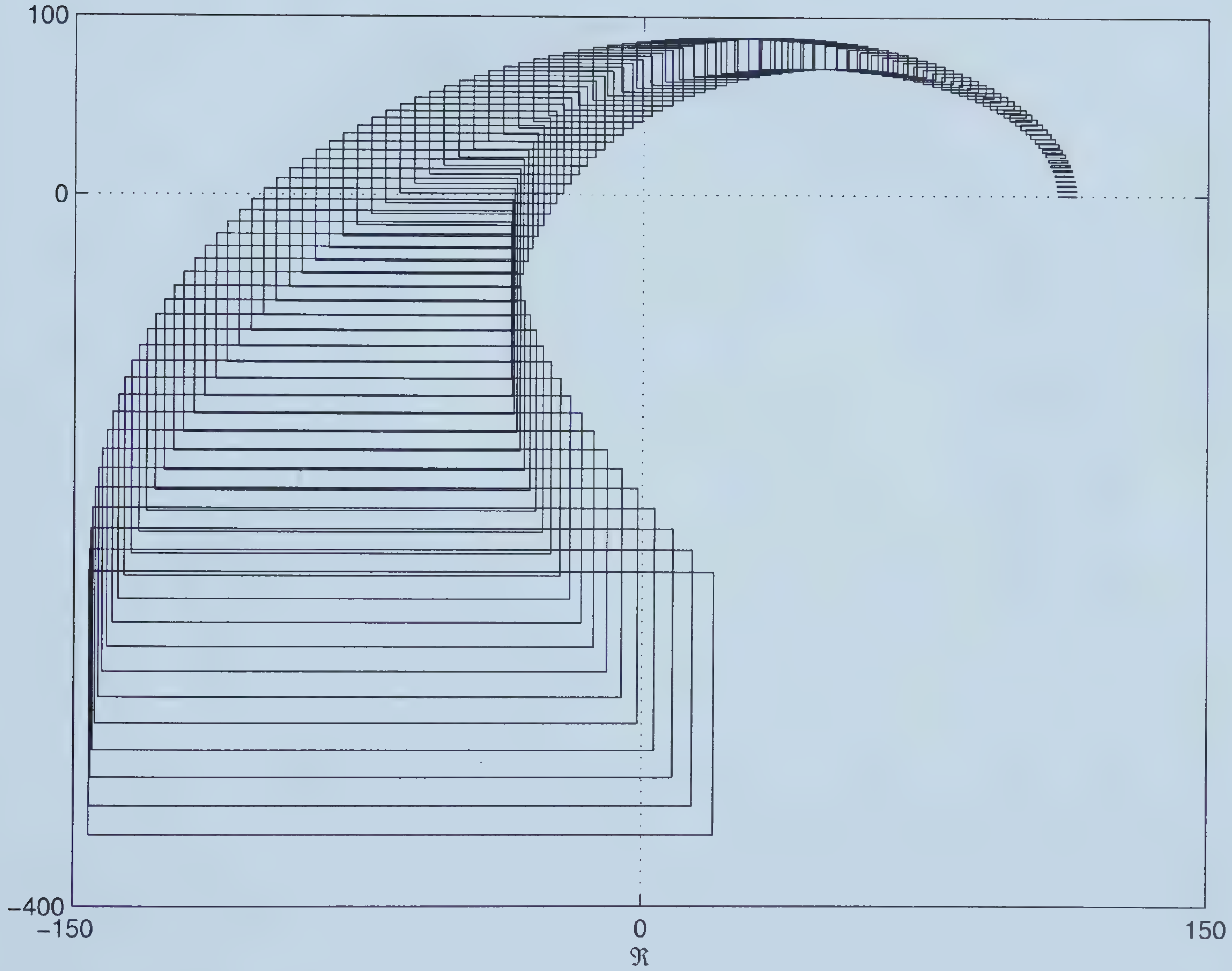

Figure 2.3: The motion of the Kharitonov rectangle for the same robustly stable interval polynomial in Fig. 2.2, but for $\omega = 0$ to $\omega = 4$.

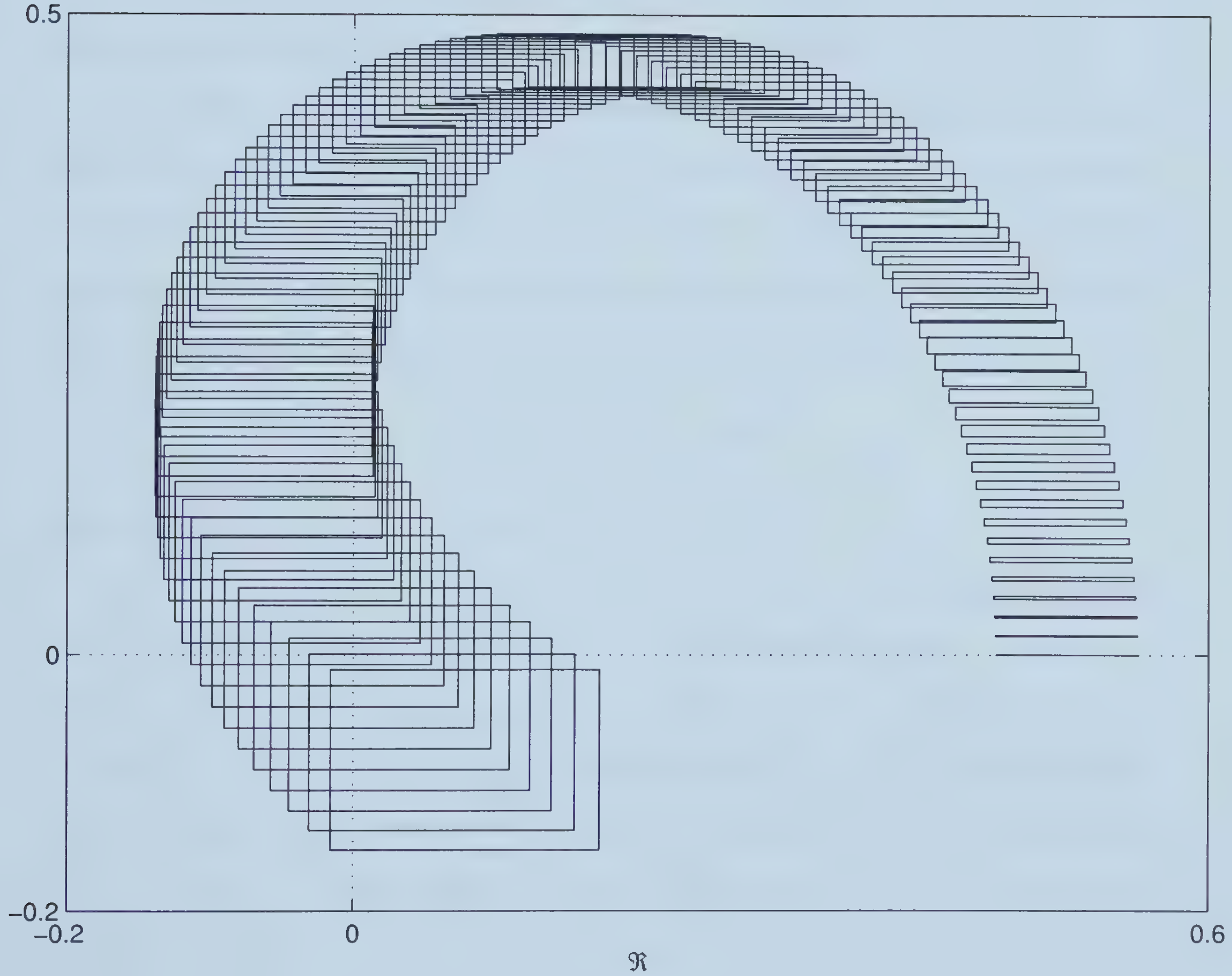

Figure 2.4: The motion of the Kharitonov rectangle for a non-robustly stable interval polynomial: $[0.45, 0.55] + [1.95, 2.05]s + [2.95, 3.05]s^2 + [5.95, 6.05]s^3 + [3.95, 4.05]s^4$, for $\omega = 0$ to $\omega = 0.75$.

Instead of checking the zero exclusion from two-dimensional Kharitonov rectangles, a scalar frequency sweeping function may be used [1]:

$$H(\omega) = \max\{\Re K_1(j\omega), -\Re K_2(j\omega), \Im K_3(j\omega), -\Im K_4(j\omega)\} \tag{2.27}$$

$\mathcal{P}$ is robustly stable if and only if $H(\omega) > 0$ for all frequencies $\omega \geq 0$. This is a frequency-domain equivalent to the Zero Exclusion Condition.

By redefining the criteria for the Zero Exclusion Condition, it can be extended to test for robust $\mathcal{D}$-stability as well. If $\mathcal{D}$ is an open subset of the complex plane and $\mathcal{P} = \{p(\cdot, q) : q \in Q\}$ is a family of polynomials with invariant degree and uncertainty bounding set Q which is pathwise connected, the coefficients $a_i(q)$ are continuous and $\mathcal{P}$ has at least one $\mathcal{D}$-stable member, then $\mathcal{P}$ is robustly $\mathcal{D}$-stable if and only if:

$$0 \notin p(z, Q), \quad \forall\, z \in \partial\mathcal{D} \tag{2.28}$$

where $\partial\mathcal{D}$ denotes the boundary of $\mathcal{D}$.

Equivalent to the frequency sweeping function $H(\omega)$ is the boundary sweeping function $\Phi_{\mathcal{D}}(\delta)$. The scalar parameter, δ, is a generalized frequency variable for $\mathcal{D}$ used to parameterize motion along boundary $\partial\mathcal{D}$ of $\mathcal{D}$. Given an interval $I \subseteq \Re$, a mapping $\Phi_{\mathcal{D}} : I \to \partial\mathcal{D}$ is a boundary sweeping function for $\mathcal{D}$ if $\Phi_{\mathcal{D}}$ is continuous and for each point $z \in \partial\mathcal{D}$, there exists some $\delta \in I$ such that $\Phi_{\mathcal{D}}(\delta) = z$. For example [33], if $\mathcal{D}$ is the half plane described by $\Re s \leq \sigma$ for some $\sigma < 0$, then the boundary sweeping function is:

$$\Phi_{\mathcal{D}}(\sigma) = \sigma + j\delta \tag{2.29}$$

2.2.2 The Edge Theorem

Robust stability can be investigated using Kharitonov rectangles, but these results are still somewhat conservative. The conservatism arises from the fact that Kharitonov rectangles usually overbound the actual value set. The rectangles result from taking a range of possible parameters and using the four extreme cases. These four extreme

points are valid in a truly independent uncertainty structure. However, real systems generally have dependent uncertainty structures, *ie.* the uncertain parameters appear in more than one coefficient, so that not all of the points within the rectangle represent possible plant models. If the uncertain polynomial is evaluated for a range of valid operating points to obtain a set of generating polynomials, or generators, $p_1(s), p_2(s), \cdots, p_n(s)$ and assume that the coefficient perturbations are polytopic instead of independent, then a polytope can be formed from the convex hull of the set of generators:

$$\mathcal{P} = \operatorname{conv}\{p_1(s), p_2(s), \cdots, p_n(s)\} \tag{2.30}$$

The polytopes, evaluated at each frequency $j\omega$, are often much smaller than the Kharitonov rectangle for the same system and frequency, and allow a less conservative control system design. A frequency sweep may be performed for each edge of the polytope to determine the cutoff frequency. The system described by the polytope is robustly stable for the frequencies in which the Zero Exclusion Condition is satisfied.

2.3 Extension to Feedback Systems

So far, we have looked at the task of applying Kharitonov's Theorem to robustness analysis of open loop systems. Now, we look at the relatively difficult task of applying it to closed loop systems. Two different approaches are investigated here. The first utilizes the Sixteen Plant Theorem and the second uses an optimal output feedback approach.

2.3.1 The Sixteen Plant Theorem

With the introduction of a fixed compensator in the feedback loop of an interval plant, the value set for the system can no longer be bounded in a rectangle. Chapellat and Bhattacharyya [10] proved that with such feedback, stability must be checked at no more than thirty-two distinguished edges to ensure robust stability. The Thirty-Two Edge Theorem is as follows [5]:

Consider an interval plant family $\mathcal{P}$ having the numerator and denominator Kharitonov polynomials $N_1(s), \cdots, N_4(s)$ and $D_1(s), \cdots, D_4(s)$ respectively, and compensator blocks $C_1(s)$ and $C_2(s)$ as shown in Fig. 2.5. Then it follows that if the family of closed loop polynomials $\mathcal{P}_{CL}$ has invariant degree, $\mathcal{P}_{CL}$ is robustly stable if and only if all edge polynomials of the form:

$$e(s,\lambda) = N_{i_1}(s)N_C(s) + D_{i_2,i_3}(s,\lambda)D_C(s) \tag{2.31}$$

with $i_1 \in \{1,2,3,4\}$ and $(i_2,i_3) \in \{(1,3),(1,4),(2,3),(2,4)\}$ or:

$$e(s,\lambda) = N_{i_1,i_2}(s)N_C(s) + D_{i_3}(s,\lambda)D_C(s) \tag{2.32}$$

with $(i_1,i_2) \in \{(1,3),(1,4),(2,3),(2,4)\}$ and $i_3 \in \{1,2,3,4\}$ are stable $\forall\lambda \in [0,1]$.

However, if the compensator is first order, then testing along these thirty-two edges is no longer required, and we can instead test sixteen distinguished cloosed loop systems. First, we need to define the sixteen Kharitonov plants:

Given an interval polynomial family P with Kharitonov polynomials $N_1(s)$, $N_2(s)$, $N_3(s)$, $N_4(s)$ and $D_1(s)$, $D_2(s)$, $D_3(s)$, $D_4(s)$ for the numerator and denominator polynomials, respectively, we define the sixteen Kharitonov plants by:

$$P_{i_1,i_2}(s) = \frac{N_{i_1}(s)}{D_{i_2}(s)} \qquad i_1, i_2 \in \{1,2,3,4\} \tag{2.33}$$

The Sixteen Plant Theorem is stated as follows [6]:

Consider the strictly proper interval plant family $\mathcal{P}$ with first order compensator $C(s) = C_1(s)C_2(s)$ as shown in Fig. 2.5. Then C(s) robustly stabilizes $\mathcal{P}$ if and only if it stabilizes each of the sixteen Kharitonov plants. In other words, the family of closed loop polynomials P_{CL} is robustly stable if and only if $P_{i_1,i_2}(s)$ is stable for $i_1, i_2 \in \{1,2,3,4\}$.

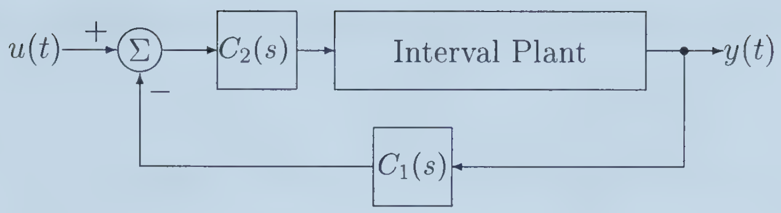

Figure 2.5: The first order compensator used with the Sixteen Plant Theorem.

This theorem can be applied to P and PI controllers in a straightforward but computationally intensive manner. One method for robustly tuning a PI controller [22] is to perform a sweep over all acceptable or desirable proportional gains, for each proportional gain solving the stabilization problem for each of the sixteen Kharitonov plants to obtain a set of corresponding integral gains. If each solution is plotted two-dimensionally, it is easy to choose from among the set of gains which robustly stabilize the interval plant family, $\mathcal{P}$. A range of stabilizing gains for a proportional controller is even more straightforward to calculate since it is a one-dimensional problem.

To obtain a set of stabilizing gains for a PID controller [22], the Sixteen Plant Theorem cannot be applied since the compensator is no longer first order. Therefore we must use a form of the Thirty-Two Edge Theorem and introduce the four Kharitonov segments of the numerator polynomial, called $NS_i(s)$, which are defined as:

$$NS_1(s) = (1-\lambda)N_1(s) + \lambda N_4(s) \tag{2.34}$$

$$NS_2(s) = (1-\lambda)N_1(s) + \lambda N_3(s) \tag{2.35}$$

$$NS_3(s) = (1-\lambda)N_4(s) + \lambda N_2(s) \tag{2.36}$$

$$NS_4(s) = (1-\lambda)N_3(s) + \lambda N_2(s) \tag{2.37}$$

where $\lambda \in [0,1]$. The sixteen Kharitonov segment plants are:

$$P_{i_1,i_2}(s,\lambda) = \frac{NS_{i_1}(s,\lambda)}{D_{i_2}(s)} \qquad i_1, i_2 \in \{1,2,3,4\} \tag{2.38}$$

If the PID controller has the transfer function $C(s) = k_p + k_i/s + k_d s$, the family of closed loop characteristic polynomials for each segment plant is denoted by $\delta_{i_1,i_2}(s, k_p, k_i, k_d)$, where:

$$\delta_{i_1,i_2}(s, k_p, k_i, k_d) = sD_{i_2}(s) + (k_i + k_p s + k_d s^2)NS_{i_1}(s, \lambda) \tag{2.39}$$

The following theorem by Ho, Datta, and Bhattacharyya is now stated for Kharitonov segment plants [22]:

> The second order compensator $C(s)$ robustly stabilizes the interval plant family $\mathcal{P}$ if and only if it stabilizes each of the sixteen Kharitonov segment plants, $P_{i_1,i_2}(s, \lambda)$, $i_1, i_2 \in \{1, 2, 3, 4\}$, $\forall \lambda \in [0, 1]$.

Now the set of all stabilizing PID controller gains can be obtained by sweeping over a range of acceptable or desired proportional gains (k_p) and solving for the stabilizing set of integral and proportional gains (k_i and k_d).

2.3.2 Simultaneous Stabilization and Optimization

The simultaneous stabilization problem is to find a single feedback controller that will stabilize each member of a collection of plants. In his monograph [11], Blondel demonstrates that it is not possible to rationally decide whether or not three or more systems is simultaneously stabilizable. However, there are testable sufficient conditions for numerous solution methods [26], which means that if they are to be used, a certain amount of design conservatism must be accepted.

There exist a variety of computational approaches to the simultaneous stabilization and optimization of a multiple plant model system [30]. Many techniques are computationally intensive because they use Kronecker products or linear matrix inequality methods. Some are subject to significant limitations, such as the minimum phase approach, which applies only to single-input transfer functions [26], and many only apply to state feedback [32]. The most intuitive techniques use gradient search methods, which involve successive solutions of algebraic Lyapunov equations. When

using gradient search methods, an important problem is that the initialization of the search process in the case of a given unstable system requires an initial stabilizing output feedback gain $\mathbf{G}$ for each plant model. This is a significant obstacle since if an initial stabilizing output feedback gain is obtained for each plant, the system is already robustly stabilized. It is also recommended that the search method allow the user some control in choosing the step size because overly large step sizes can lead to feedback gains which destabilize one or more of the plant model systems and thus lead to failures because of unavailable objective function or gradient values. One drawback to gradient search techniques is that they typically only find a local minimum to the objective function, and not a global minimum [35]. It is suggested [9] that an appropriate solution method is the BFGS quasi-Newton algorithm. It is a limited memory algorithm for solving large nonlinear optimization problems with simple bounds on the variables [15] [38]. There are numerous other techniques available [31] [36] [34], however, the algorithm by Broussard and McLean presented in [13] was chosen for this work because it solves the initial stabilizing gain problem by using any initial stabilizing full state feedback gain and using equality contraints to ensure that an equivalent output feedback gain exists [12]. The algorithm also adjusts the step size dynamically. For brevity, their algorithm will be referred to simply as the multiple model algorithm.

Multiple Model Algorithm

Assume there are N plant models, each of the form:

$$\dot{\mathbf{x}} = \mathbf{A}_j\mathbf{x} + \mathbf{B}_j\mathbf{u} \tag{2.40}$$

$$\mathbf{y} = \mathbf{C}_j\mathbf{x} \tag{2.41}$$

where $i = 1, \cdots, N$, $\mathbf{x} \in \Re^n$, $\mathbf{u} \in \Re^m$, and $\mathbf{y} \in \Re^p$. Note that the general form of Eq. 2.41 is $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$, but without loss of generality we assume $\mathbf{D} = 0$. It is easy to see that if $\mathbf{D} \neq 0$ and $\mathbf{F}$ is the solution of the above problem for $\mathbf{D}$ set to zero, then

the solution for nonzero $\mathbf{D}$ is $\mathbf{K} = (\mathbf{I} + \mathbf{FD})^{-1}\mathbf{F}$, provided that $\mathbf{I} + \mathbf{FD}$ is invertible.

Let $\bar{\mathbf{R}}$ be the right inverse and $\mathbf{V}$ be the right annihilator of the plant output measurement matrix $\mathbf{C}$. Assume that $\mathbf{C}$ is full rank so that $\bar{\mathbf{R}}$ and $\mathbf{V}$ can be calculated from:

$$\bar{\mathbf{R}} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} \tag{2.42}$$

$$\mathbf{V} = \mathbf{I} - \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C} = \mathbf{I} - \bar{\mathbf{R}}\mathbf{C} \tag{2.43}$$

Assume that $\mathbf{K}$ is a full state feedback gain matrix that satisfies:

$$\mathbf{K}\mathbf{V} = 0 \tag{2.44}$$

Then it follows that:

$$\mathbf{K}(\mathbf{I} - \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C}) = 0 \tag{2.45}$$

Multiplying the $\mathbf{K}$ through and substituting $\bar{\mathbf{R}}$:

$$\begin{aligned} \mathbf{K} &= [\mathbf{K}\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}]\mathbf{C} && (2.46) \\ &= [\mathbf{K}\bar{\mathbf{R}}]\mathbf{C} && (2.47) \\ &= \mathbf{G}\mathbf{C} && (2.48) \end{aligned}$$

Which gives us our definition for the output feedback gain, $\mathbf{G}$:

$$\mathbf{G} = \mathbf{K}\bar{\mathbf{R}} \tag{2.49}$$

In other words, if a full state feedback gain satisfies $\mathbf{KV} = 0$, then an equivalent output feedback gain exists as $\mathbf{G} = \mathbf{K}\bar{\mathbf{R}}$. It will suffice to check that:

$$\mathbf{K}\bar{\mathbf{V}} = 0 \tag{2.50}$$

where $\bar{\mathbf{V}}$ is the null space of $\mathbf{C}$. $\bar{\mathbf{V}}$ consists of a set of k column vectors, $\mathbf{v}_j$, so that Eq. 2.50 can be broken up into $m \times k$ scalar equations. Define $\mathbf{r}$ as the $m \times m$ identity matrix, $\mathbf{I}_m$, repeated k times:

$$\mathbf{r} = \begin{bmatrix} \mathbf{I}_{m_1} \\ \vdots \\ \mathbf{I}_{m_k} \end{bmatrix} \tag{2.51}$$

Also define $\mathbf{v}$ to be each column of $\bar{\mathbf{V}}$ repeated m times:

$$\mathbf{v} = \left[\begin{array}{ccccccccc} \bar{\mathbf{V}}(:,1)_1 & \cdots & \bar{\mathbf{V}}(:,1)_m & \bar{\mathbf{V}}(:,2)_1 & \cdots & \bar{\mathbf{V}}(:,k)_1 & \cdots & \bar{\mathbf{V}}(:,k)_m \end{array} \right] \quad (2.52)$$

And also for convenience, let $\mathbf{r}_i$ refer to the i^{th} row of $\mathbf{r}$ and let $\mathbf{v}_i$ refer to the i^{th} column of $\mathbf{v}$. Now, Eq. 2.50 is equivalent to:

$$0 = \mathbf{r}_i \mathbf{K} \mathbf{v}_i \qquad i = 1, \cdots, m \times k \quad (2.53)$$

This is equivalent to taking each individual row of $\mathbf{K}$ times each column of $\bar{\mathbf{V}}$ and equating it to a scalar zero, which forces certain certain linear combinations of individual gain elements to zero while the rest of the gains are used to optimize the objective function. Therefore, by enforcing the equality constraints, Eq. 2.53, on our full state feedback gain, $\mathbf{K}$, we ensure the existence of an equivalent output feedback gain, $\mathbf{G}$.

For a given gain matrix $\mathbf{K}_j$, a quadratic objective function can be written:

$$J_j = \int_0^\infty (\mathbf{x}_j^T \mathbf{Q}_j \mathbf{x}_j + \mathbf{u}_j^T \mathbf{R}_j \mathbf{u}_j) dt \quad (2.54)$$

which has the well known solution [25]:

$$J_j = \mathrm{tr}\{\mathbf{P}_j \mathbf{x}_0 \mathbf{x}_0^T\} \quad (2.55)$$

where $\mathbf{x}_{0_j}$ is the initial condition of $\mathbf{x}_j$ and $\mathbf{P}_j \geq 0$ is the solution of:

$$W(\mathbf{K}_j, \mathbf{P}_j) = \tilde{\mathbf{A}}_j^T \mathbf{P}_j + \mathbf{P}_j \tilde{\mathbf{A}}_j + \mathbf{Q}_j + \mathbf{K}_j^T \mathbf{R}_j \mathbf{K}_j = 0 \quad (2.56)$$

The accent $\tilde{}$ is used to denote a closed loop quantity, in other words, $\tilde{\mathbf{A}} = \mathbf{A} - \mathbf{BK}$. The dependence of Eq.2.55 on initial conditions can be eliminated by assuming that $\mathbf{x}_0$ is randomly distributed and the problem is now modified to that of minimizing $E\{J\}$, where E is the expectation operator. This is done by replacing $\mathbf{x}_0 \mathbf{x}_0^T$ by $\tilde{\mathbf{X}}$, the covariance of the expected initial condition distribution. Assuming that the states are uncorrelated results in $\tilde{\mathbf{X}}_j = \mathbf{I}_n$. The individual objective functions, J_j, are now:

$$J_j = \mathrm{tr}\{\mathbf{P}_j \tilde{\mathbf{X}}_j\} + \mathrm{tr}\{W(\mathbf{K}_j, \mathbf{P}_j) \mathbf{S}_j^T\} \quad (2.57)$$

where $\mathbf{S}_j$ satisfies the algebraic Lyapunov equation:

$$\tilde{\mathbf{A}}_j\mathbf{S}_j + \mathbf{S}_j\tilde{\mathbf{A}}_j^T + \tilde{\mathbf{X}}_j = 0 \tag{2.58}$$

The overall objective function will simply be the sum of the individual objective functions, J_j:

$$J = \sum_{j=1}^{N} J_j \tag{2.59}$$

The equality constraints, Eq. 2.53, will be enforced explicitly for model 1, and will be enforced implicitly for the remaining models by utilizing our definition of $\mathbf{G}$, Eq. 2.49, as additional constraints:

$$\begin{aligned} \mathbf{K}_j &= \mathbf{G}\mathbf{C}_j && (2.60)\\ &= \mathbf{K}_1\bar{\mathbf{R}}_1\mathbf{C}_j && (2.61)\\ &= \mathbf{K}_1\bar{\mathbf{C}}_j \qquad j = 2,\cdots N && (2.62)\end{aligned}$$

where $\bar{\mathbf{C}}_j \triangleq \mathbf{C}_1^T(\mathbf{C}_1\mathbf{C}_1^T)^{-1}\mathbf{C}_j$.

In order to solve for the equality constraints, we will adjoin the constraints with the performance index, J, by a set of "unknown multipliers," $\{L_i \mid i = 1,\cdots,m\times k\}$, called Lagrange multipliers [14]. Now the quadratic objective function, Eq. 2.59, becomes:

$$J = \sum_{j=1}^{N} J_j + \sum_{i=1}^{m\times k} \mathbf{r}_i\mathbf{K}_1\mathbf{v}_iL_i \tag{2.63}$$

The constraints for the remaining models will also be adjoined to the quadratic objective function using a set of Lagrange multipliers, $\{\tilde{\mathbf{X}}_j \mid j = 2,\cdots,N\}$. Then the quadratic objective function in its final form is:

$$J = \sum_{j=1}^{N} J_j + \sum_{i=1}^{m\times k} \mathbf{r}_i\mathbf{K}_1\mathbf{v}_iL_i + \mathrm{tr}\left\{\sum_{j=2}^{N}(\mathbf{K}_j - \mathbf{K}_1\bar{\mathbf{C}}_j)\mathbf{X}_j^T\right\} \tag{2.64}$$

The minimization of Eq. 2.64 requires finding a *stationary value* of J. A stationary point of J occurs when $dJ = 0$ for arbitrary input variations, which gives us the following necessary conditions [13]:

$$\frac{\partial J}{\partial S_j} = \tilde{\mathbf{A}}_j^T\mathbf{P}_j + \mathbf{P}_j\tilde{\mathbf{A}}_j + \mathbf{Q}_j + \mathbf{K}_j^T\mathbf{R}_j\mathbf{K}_j = 0 \qquad j = 1,\cdots,N \tag{2.65}$$

$$\frac{\partial J}{\partial P_j} = \tilde{\mathbf{A}}_j\mathbf{S}_j + \mathbf{S}_j\tilde{\mathbf{A}}_j^T + \tilde{\mathbf{X}}_j = 0 \qquad j = 1, \cdots, N \tag{2.66}$$

$$\frac{\partial J}{\partial L_j} = \mathbf{r}_i\mathbf{K}_1\mathbf{v}_i = 0 \qquad i = 1, \cdots, m \times k \tag{2.67}$$

$$\frac{\partial J}{\partial X_j} = \mathbf{K}_j - \mathbf{K}_1\bar{\mathbf{C}}_j = 0 \qquad j = 2, \cdots, N \tag{2.68}$$

$$\frac{\partial J}{\partial K_j} = \mathbf{R}_j\mathbf{K}_j\mathbf{S}_j - \mathbf{B}_j^T\mathbf{P}_j\mathbf{S}_j + \mathbf{X}_j = 0 \qquad j = 2, \cdots, N \tag{2.69}$$

$$\frac{\partial J}{\partial K_1} = \mathbf{R}_1\mathbf{K}_1\mathbf{S}_1 - \mathbf{B}_1^T\mathbf{P}_1\mathbf{S}_1 + \sum_{i=1}^{m\times k}\mathbf{r}_i^T L_i\mathbf{v}_i^T - \sum_{j=2}^{N}\mathbf{X}_j\bar{\mathbf{C}}_j^T = 0 \tag{2.70}$$

The algorithm is as follows [13]:

Step 0.

Choose $\mathbf{K}_j \in S_{uj}$, where:

$$S_{uj} = \{\mathbf{K}_j \mid \mathbf{A}_j - \mathbf{B}_j\mathbf{K}_j \text{ is Hurwitz}\} \qquad j = 1, \cdots, N \tag{2.71}$$

Set the algorithm increment to zero ($p = 0$) and *state* $= 1$.

Step 1.

Solve the algebraic Lyapunov equations for the necessary conditions Eq. 2.65 and Eq. 2.66:

$$\tilde{\mathbf{A}}_{jp}^T\mathbf{P}_{jp} + \mathbf{P}_{jp}\tilde{\mathbf{A}}_{jp} + \mathbf{Q}_j + \mathbf{K}_{jp}^T\mathbf{R}_j\mathbf{K}_{jp} = 0 \tag{2.72}$$

$$\tilde{\mathbf{A}}_{jp}\mathbf{S}_{jp} + \mathbf{S}_{jp}\tilde{\mathbf{A}}_{jp}^T + \tilde{\mathbf{X}}_j = 0 \tag{2.73}$$

Step 2.

Apply the equality constraints, which are the necessary condition Eq. 2.67, by solving for $\mathbf{X}_j$, $j = 2, \cdots, N$ and the scalars L_{ip}, $i = 1, \cdots, m \times k$, using:

$$\mathbf{X}_{jp} = \mathbf{B}_j^T\mathbf{P}_{jp}\mathbf{S}_{jp} - \mathbf{R}_j\mathbf{K}_{jp}\mathbf{S}_{jp} \tag{2.74}$$

$$L_p = \mathbf{Z}\mathbf{Y}^{-1} \tag{2.75}$$

where:

$$L_p = \begin{bmatrix} L_{1p} & L_{2p} & \cdots & L_{(m\times k)p} \end{bmatrix} \tag{2.76}$$

$$Y_{xy} = \mathbf{r}_y\mathbf{R}_1^{-1}\mathbf{r}_x^T\mathbf{S}_{1p}^{-1}\mathbf{v}_y \qquad x, y = 1, \cdots, m \times k \tag{2.77}$$

$$Z_y = \mathbf{r}_y\mathbf{R}_1^{-1}\left[\mathbf{B}_1^T\mathbf{P}_{1p}\mathbf{S}_{1p} + \sum_{j=2}^{N}\mathbf{X}_{jp}\bar{\mathbf{C}}_j^T\right]\mathbf{S}_{1p}^{-1}\mathbf{v}_y \qquad y = 1,\cdots,m\times k \tag{2.78}$$

Y_{xy} denotes individual elements of the $N \times N$ matrix $\mathbf{Y}$ and Z_y denotes individual elements of the $1 \times N$ row vector $\mathbf{Z}$.

Step 3.

Compute the gradient for model 1.

$$\Delta\mathbf{K}_1 = \mathbf{R}_1^{-1}\left[\mathbf{B}_1^T\mathbf{P}_{1p}\mathbf{S}_{1p} + \sum_{j=2}^{N}\mathbf{X}_{jp}\bar{\mathbf{C}}_j^T - \sum_{i=1}^{m\times k}\mathbf{r}_i^T L_{ip}\mathbf{v}_i^T\right]\mathbf{S}_{1p}^{-1} - \mathbf{K}_{1p} \tag{2.79}$$

Define:

$$\bar{\mathbf{K}}_1 = \mathbf{R}_1^{-1}\left[\mathbf{B}_1^T\mathbf{P}_{1p}\mathbf{S}_{1p} + \sum_{j=2}^{N}\mathbf{X}_{jp}\bar{\mathbf{C}}_j^T - \sum_{i=1}^{m\times k}\mathbf{r}_i^T L_{ip}\mathbf{v}_i^T\right]\mathbf{S}_{1p}^{-1} \tag{2.80}$$

$$\Delta\mathbf{K}_1 = \bar{\mathbf{K}}_1 - \mathbf{K}_{1p} \tag{2.81}$$

Step 4.

If in state 1, compute the gradient for the remaining models using the necessary condition Eq. 2.68:

$$\begin{aligned}\Delta\mathbf{K}_{jp} &= \bar{\mathbf{K}}_{1p}\bar{\mathbf{C}}_j - \mathbf{K}_{jp}\\ &= \bar{\mathbf{K}}_{jp} - \mathbf{K}_{jp} \qquad j = 2,\cdots,N\end{aligned} \tag{2.82}$$

The next step gains will be calculated using:

$$\mathbf{K}_{j\ p+1} = \mathbf{K}_{jp} + \alpha\Delta\mathbf{K}_{jp} \qquad \forall j \tag{2.83}$$

Now select a step size, $\alpha \in (0,1]$, such that $\mathbf{K}_{j\ p+1} \in S_{uj}\forall j$, in other words, select a step size such that the next step gains stabilize their closed loop systems: $\mathbf{A}_j - \mathbf{B}\mathbf{K}_j$. If $\alpha = 1$ can be used, then we know:

$$\begin{aligned}\mathbf{K}_{1\ p+1} \in &\ \{\ \mathbf{K} \in S_{u1} \mid \mathbf{r}_i\mathbf{K}\mathbf{v}_i = 0,\ i = 1,\cdots,m\times k\}\\ \cup &\ \{\ \mathbf{K}\bar{\mathbf{C}}_j \in S_{uj},\ j = 2,\cdots,N\}\end{aligned} \tag{2.84}$$

because we have satisfied the necessary condition Eq. 2.68. In other words, if $\alpha = 1$ can be used, then an initial stabilizing gain (Kharitonov gain), $\mathbf{G}$ has been found for all models, and we set $state = 2$ and goto Step 6.

Step 5.

If in state 2, search amongst the Kharitonov gains to find a local minimum to the objective function. Compute the next step gains as:

$$\mathbf{K}_{1\ p+1} = \mathbf{K}_{1p} + \alpha \Delta \mathbf{K}_{1p} \tag{2.85}$$

$$\mathbf{K}_{j\ p+1} = \mathbf{K}_{1\ p+1} \bar{\mathbf{C}}_j \qquad j = 2, \cdots, N \tag{2.86}$$

Select a step size, $\alpha \in (0, 1]$, such that $J(\mathbf{K}_{1\ p+1}) < J(\mathbf{K}_{1p})$. In other words, select the step size so that the objective function decreases in value.

Step 6.

Set $p = p + 1$ and goto Step 1.

A MATLAB script file for this algorithm using four plants, called `simstab4.m`, is shown in Appendix A.4.

An effective modification to the algorithm allows the addition of extra plants to add weight to the optimization for models representing specific operating points. For example, a fifth plant could be added to add weight to the plant model representing the nominal operating condition. All that is required is to modify the quadratic objective function, Eq. 2.64, accordingly. For example, it is probably desirable to optimize the plant for the nominal operating condition and give little regard to optimizing the four Kharitonov plants, which are extreme plants, so the quadratic objective function could be modified so that the nominal plant optimal gain is adjusted to just barely stabilize all the models:

$$\sum_{j=1}^{N} J_j = J_{nominal} + 0.001 \sum_{j=1}^{4} J_{Kharitonov_j} \tag{2.87}$$

A MATLAB script file for this five plant algorithm, called `simstab5.m`, is shown in Appendix A.5.

If states are measurable, a multivariable state feedback controller can be used instead of an output feedback controller. When employing the multiple model algorithm, $\mathbf{C} = \mathbf{I}_n$ is not allowable since the null space of $\mathbf{I}_n$ is an empty matrix.

This necessitates some modifications to the algorithm. First, eliminate the equality constraint, Eq. 2.53, in the quadratic objective function, Eq. 2.64. This equality constraint ensures the existence of an output feedback gain equivalent to the constrained state feedback gain, and is therefore unnecesary for a full state feedback controller. Therefore, the quadratic objective function becomes:

$$J = \sum_{j=1}^{N} J_j + \mathrm{tr}\left\{ \sum_{j=2}^{N} (\mathbf{K}_j - \mathbf{K}_1 \bar{\mathbf{C}}_j)\mathbf{X}_j^T \right\} \tag{2.88}$$

In Step 2, it is only necessary to solve for $\mathbf{X}_j$, $j = 2, \cdots, N$, and in Step 3, the definition of $\bar{\mathbf{K}}_1$ (Eq. 2.80) is now changed to:

$$\bar{\mathbf{K}}_1 = \mathbf{R}_1^{-1} \left[\mathbf{B}_1^T \mathbf{P}_{1p} \mathbf{S}_{1p} + \sum_{j=2}^{N} \mathbf{X}_{jp} \bar{\mathbf{C}}_j^T \right] \mathbf{S}_{1p}^{-1} \tag{2.89}$$

The gradient for model 1 is still calculated as in Eq. 2.81. Step 4 is eliminated since it is no longer necessary to search for a set of initial stabilizing output feedback gains. A MATLAB script file for the full state feedback algorithm, called `simstab4sf.m`, is shown in Appendix A.6.

[illegible]

Chapter 3

Steam System Dynamic Modeling

As in most modern control theory, the controller designs presented in Chapter 4 require mathematical models of the process to be controlled. A first principles model is derived for this case study, and a procedure for obtaining system identification models is presented for the cases where a first principles model is impractical.

3.1 Overview of Syncrude's Steam Header System

3.1.1 900# Header

A closed loop, or ring header, collects the 900# steam from the three utility boilers, two CO boilers, and two once-through steam generators and distributes it to numerous drivers and letdowns, as shown in Fig. 3.1. The regulatory and protective controls on the 900# header are summarized as follows:

- The pressure setpoint, nominally 915 psig, is modulated by three utility boilers, each with a maximum continuous rating of 800 kpph.

- A high pressure override at 930 psig forces the 50# atmospheric vent open, and drawdown is through the 900/600# and the 600/50# letdowns.

- A directional block prevents the 600/50# valves from opening once the 900# header pressure drops below 907 psig.

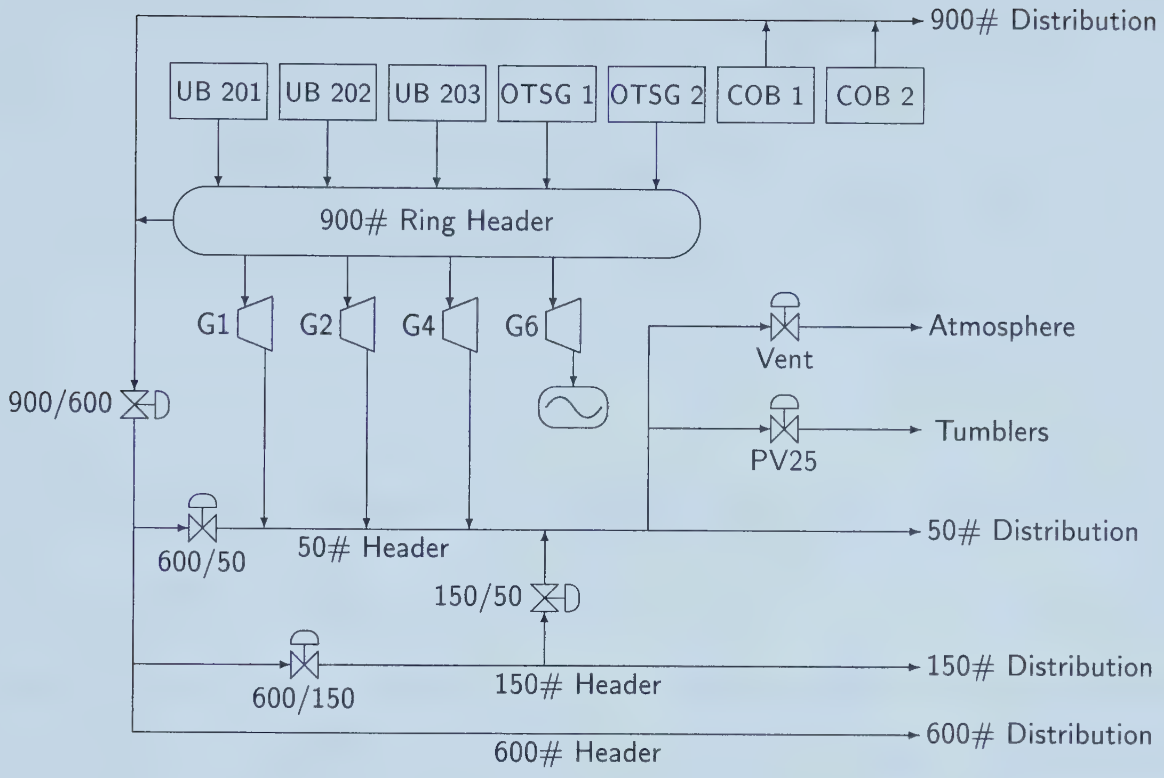

Figure 3.1: A simplified process flow diagram of the steam header system.

- At 905 psig, initial pressure limiters (IPL's) begin to unload the backpressure steam turbine generators (G1, G2, and G4).
- At 900 psig, an IPL begins to unload the condensing steam turbine generator (G6).
- At 890 psig, the CO boilers trip from flow control to manual control.
- When the total utility boiler steam flow is at the peak limiter setpoint, directional blocks prevent load increases on the backpressure steam turbine generators (G1, G2, and G4), and an additional directional block prevents further 600/50# letdown opening.
- When the total utility boiler steam flow is 50 kpph less than the peak limiter setpoint, a directional block prevents the condensing steam turbine generator (G6) from increasing its load when it is in automatic control.

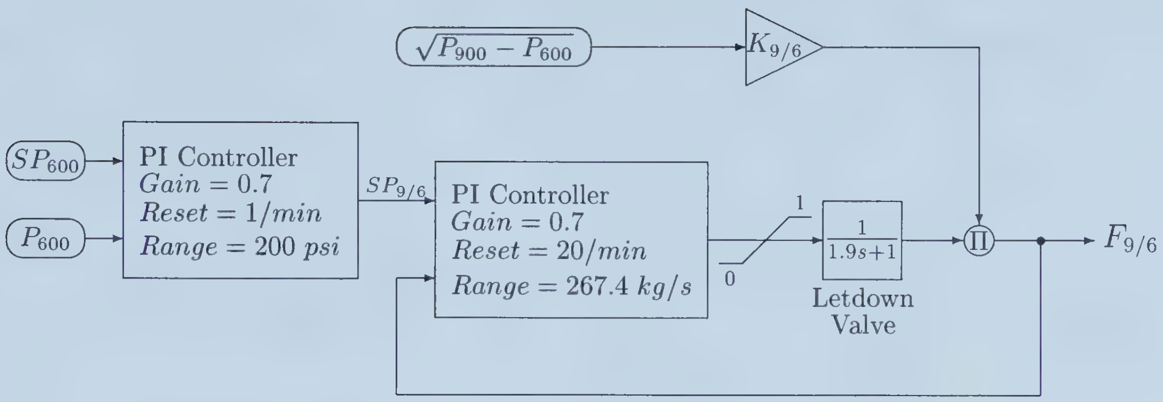

Figure 3.2: Block diagram of the existing 900/600# letdown control system.

3.1.2 600# Header

Most of the 600# steam is generated and distributed internally within the upgrader by waste heat steam generators. This is usually sufficient to maintain their own needs and provide a small export to the utility system. The main uses for this steam are to supply medium to large steam turbine drivers as well as to provide steam for process heating, atomization, and fluidization. The regulatory and protective controls are as follows:

- The 900/600# letdowns modulate pressure to a setpoint of 615 psig.
- An overpressure controller forces the 600/50# letdown to start opening when the 600# pressure rises to 623 psig.
- An underpressure override at 607 psig starts runback on the 600/50# letdown.

3.1.3 150# Header

150# steam comes either from the low pressure side of various steam turbine drivers, or from waste heat steam generators. Uses for this steam include providing stripping steam for the fluid cokers, steam heat tracing, process and building heating, and drive steam for small turbines. Tight pressure control is maintained quite simply through

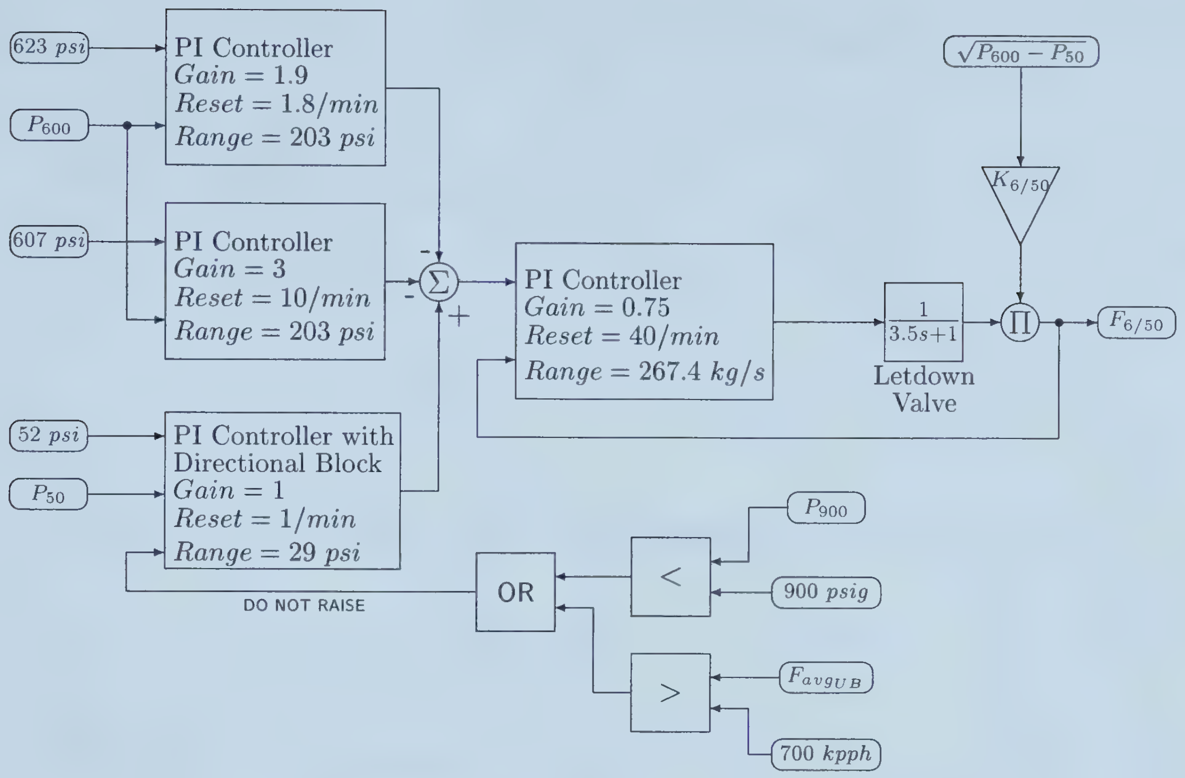

Figure 3.3: Block diagram of the existing 600/50# letdown control system and its various overrides.

the following controls:

- The setpoint of 155 psig is modulated by the 600/150# letdown.
- If the pressure reaches 157.5 psig, an overpressure controller forces open the 150/50# letdowns.

3.1.4 50# Header

Most 50# steam is provided by the low pressure side of steam turbines. This steam is used for heating buildings, water, ore tumblers, deaerators, amine units, and diluent recovery.

- When in automatic control, the backpressure steam turbine generators (G1, G2, and G4) maintain a setpoint of 54 psig.

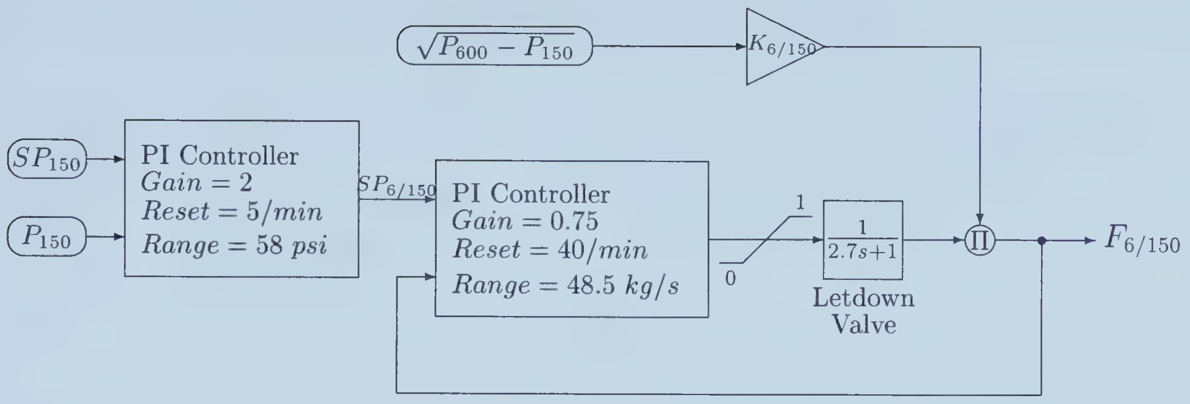

Figure 3.4: Block diagram of the existing 600/150# letdown control system.

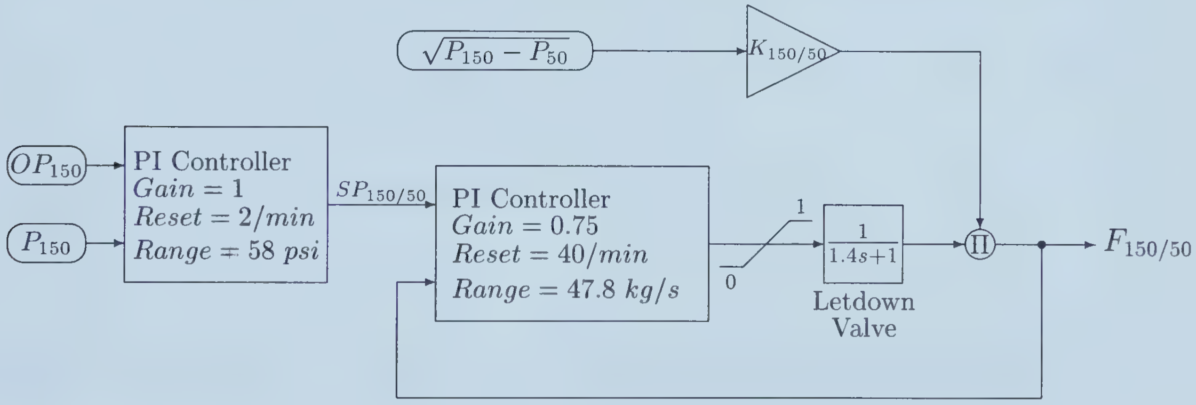

Figure 3.5: Block diagram of the existing 150/50# letdown control system. Remember that this is an overpressure controller, so OP_{150} is the overpressure setpoint and not the header pressure setpoint (SP_{150}).

- The 600/50# letdown has a setpoint of 52 psig. It normally responds to low 50# pressure if the backpressure turbines can't respond quickly enough.
- A high pressure override at 58 psig opens the 50# atmospheric vents.
- A low pressure override at 48 psig throttles a valve to restrict the ore tumbler load in the Extraction Department.

3.2 Simplified Steam System Dynamics

The continuity equation for a steam vessel [8] says that the mass flow into the vessel minus the mass flow out of the vessel is equal to the time rate of change of mass inside the vessel, or mathematically:

$$F_{in} - F_{out} = \frac{dM}{dt} = V\frac{d\rho}{dt} \tag{3.1}$$

where:

$$\begin{aligned}
M &= V\rho = \text{mass of steam in the vessel (kg)} \\
V &= \text{volume of vessel (m}^3\text{)} \\
\rho &= \text{density of steam (kg/m}^3\text{)} \\
F &= \text{steam mass flow rate (kg/s)} \\
t &= \text{time (s)}
\end{aligned}$$

Assuming that the flow out of the vessel is proportional to the pressure in the vessel:

$$F_{out} = \frac{F_0}{P_0}P \tag{3.2}$$

$$P = F_{out}\frac{P_0}{F_0} \tag{3.3}$$

$$\frac{dP}{dt} = \frac{P_0}{F_0}\frac{dF_{out}}{dt} \tag{3.4}$$

where:

$$\begin{aligned}
P &= \text{pressure of steam in the vessel (MPa)} \\
P_0 &= \text{rated pressure} \\
F_0 &= \text{rated flow out of vessel}
\end{aligned}$$

Assuming that temperature within the vessel is constant:

$$\frac{d\rho}{dt} = \frac{dP}{dt}\frac{\partial\rho}{\partial P} \tag{3.5}$$

$\frac{\partial \rho}{\partial P}$ can be obtained from steam tables or thermodynamic calculations. Putting together Eq. 3.1, Eq. 3.4, and Eq. 3.5:

$$\begin{aligned} F_{in} - F_{out} &= V \frac{dP}{dt} \frac{\partial \rho}{\partial P} && (3.6) \\ &= V \frac{\partial \rho}{\partial P} \frac{P_0}{F_0} \frac{dF_{out}}{dt} && (3.7) \\ &= \tau_v \frac{dF_{out}}{dt} && (3.8) \end{aligned}$$

where $\tau_v = V \frac{P_0}{F_0} \frac{\partial \rho}{\partial P}$ is the time constant of the vessel. Now, if we put Eq. 3.8 into Laplace form:

$$\begin{aligned} F_{in} - F_{out} &= s\tau_v F_{out} \\ \frac{F_{out}}{F_{in}} &= \frac{1}{\tau_v s + 1} && (3.9) \end{aligned}$$

Note that in deriving a very commonly used linear model for steam flow, we have made several assumptions, the most questionable being:

- The flow out of the vessel is proportional to the pressure in the vessel. This is questionable because even when using this equation as a dynamic element, the actual flow calculations will of course use the square root of the difference between the vessel pressure and the downstream pressure.

Assumptions such as this are our motivation to use robust control theory.

The equation for a servo steam valve is derived in a similar fashion, but uses the assumptions that the steam valves have linearizing compensation and that the flow of steam is proportional to the valve position and the pressure across the valve. The equation for a steam valve is simply a first-order model for a servomechanism:

$$\frac{valve\ position}{setpoint} = \frac{1}{\tau_{valve} s + 1} \quad (3.10)$$

Now that simple flow dynamics have been established, we can use the same procedure to come up with pressure dynamics. Starting from Eq. 3.6:

$$F_{in} - F_{out} = V \frac{dP}{dt} \frac{\partial \rho}{\partial P} \quad (3.11)$$

Let $C = V\frac{\partial\rho}{\partial P}$ = capacitance of the vessel in kg/MPa, then:

$$F_{in} - F_{out} = C\frac{dP}{dt} \tag{3.12}$$

$$P = \frac{1}{C}\int(F_{in} - F_{out})dt \tag{3.13}$$

$$P = \frac{1}{C}\frac{F_{in} - F_{out}}{s} \tag{3.14}$$

The capacities can be estimated by using an appropriate pressure range to approximate $\frac{\partial\rho}{\partial P}$:

$$\frac{\partial\rho_{900\#}}{\partial P_{900\#}} \approx \frac{\rho_{965\ psia} - \rho_{865\ psia}}{965\ psia - 865\ psia} = 3.119\ \frac{kg/m^3}{MPa} \tag{3.15}$$

$$\frac{\partial\rho_{600\#}}{\partial P_{600\#}} \approx \frac{\rho_{655\ psia} - \rho_{605\ psia}}{655\ psia - 605\ psia} = 3.653\ \frac{kg/m^3}{MPa} \tag{3.16}$$

$$\frac{\partial\rho_{150\#}}{\partial P_{150\#}} \approx \frac{\rho_{195\ psia} - \rho_{145\ psia}}{195\ psia - 145\ psia} = 4.776\ \frac{kg/m^3}{MPa} \tag{3.17}$$

$$\frac{\partial\rho_{50\#}}{\partial P_{50\#}} \approx \frac{\rho_{80\ psia} - \rho_{50\ psia}}{80\ psia - 50\ psia} = 4.451\ \frac{kg/m^3}{MPa} \tag{3.18}$$

Then the capacities are calculated by substituting the values into the equation $C = V\frac{\partial\rho}{\partial P}$. The volumes shown below were calculated from Bechtel, Inc. design data [7] and the densities were taken from steam tables [23].

$$V_{900\#} = 142.4\ m^3 \tag{3.19}$$

$$V_{600\#} = 369.4\ m^3 \tag{3.20}$$

$$V_{150\#} = 409.2\ m^3 \tag{3.21}$$

$$V_{50\#} = 3437\ m^3 \tag{3.22}$$

Now it is simple to calculate the capacities in kg/MPa:

$$C_{900} = 444.1\ kg/MPa \tag{3.23}$$

$$C_{600} = 1349\ kg/MPa \tag{3.24}$$

$$C_{150} = 1954\ kg/MPa \tag{3.25}$$

$$C_{50} = 1.530 \times 10^4\ kg/MPa \tag{3.26}$$

3.2.1 Steam System Model

The pressure controllers are not modeled because the control systems designed using this model will keep the existing flow controllers, but will replace the pressure controllers in the cascade loops which give the setpoints to the flow controllers.

The states for the $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$ model are the four header pressure deviations and the four letdown station flow controller integrator states:

$$\begin{aligned}
x_1 &= P_{900} - 915\ psig \\
x_2 &= P_{600} - 615\ psig \\
x_3 &= P_{150} - 155\ psig \\
x_4 &= P_{50} - 55\ psig \\
x_5 &= 900/600\#\ integrator\ state \\
x_6 &= 600/150\#\ integrator\ state \\
x_7 &= 600/50\#\ integrator\ state \\
x_8 &= 150/50\#\ integrator\ state
\end{aligned}$$

The control inputs for are the letdown station flow setpoints:

$$\begin{aligned}
u_1 &= SP_{900/600\#\ letdown\ flow} \\
u_2 &= SP_{600/150\#\ letdown\ flow} \\
u_3 &= SP_{600/50\#\ letdown\ flow} \\
u_4 &= SP_{150/50\#\ letdown\ flow}
\end{aligned}$$

Again, the header pressure fluctuations are modeled simply as a relationship between the capacity of the header (in kg/MPa) and the modeled flows in and out of the header.

$$\dot{x}_1 = \frac{1}{444.1}(-F_{9/6}) \tag{3.27}$$

$$\dot{x}_2 = \frac{1}{1349}(F_{9/6} - F_{6/150} - F_{6/50}) \tag{3.28}$$

$$\dot{x}_3 = \frac{1}{1954}(F_{6/150} - F_{150/50}) \tag{3.29}$$

$$\dot{x}_4 = \frac{1}{1.530 \times 10^4}(F_{6/50} + F_{150/50}) \tag{3.30}$$

Parametric uncertainties arise from the linearization of the flow equation, $F = K\sqrt{\Delta P} \times IVP$, where K is a flow constant, ΔP is the differential pressure, and IVP is the implied valve position. This equation is crudely linearized by replacing $K\sqrt{\Delta P}$ with an uncertainty reflecting a range of realistic pressures. Now it is straightforward to derive equations for the integrator states. Start with the equation for the serial form of a PI controller:

$$CV = Gain \times (1 + Reset \times \frac{1}{s}) \times Error \tag{3.31}$$

If we make x_{int} the state associated with the integrator, then we can put this into a more useful form:

$$CV = x_{int} + \frac{1}{Reset}\dot{x}_{int} \tag{3.32}$$

$$\dot{x}_{int} = \frac{1}{Range} \times Gain \times Error \tag{3.33}$$

Using this for the 900/600# controller, and letting the flow be a range of possible flows:

$$F_{9/6} = Flow \times (x_5 + \frac{1}{Reset}\dot{x}_5) = [244.0, 288.7](x_5 + 3\dot{x}_5) \tag{3.34}$$

where x_5 is the integrator state.

$$\dot{x}_5 = Gain \times Reset(u_1 - F_{9/6}/Range) \tag{3.35}$$

$$= 0.2333\,(u_1 - [0.9125, 1.080](x_5 + 3\dot{x}_5)) \tag{3.36}$$

The *Gain*, *Reset*, and *Range* values are shown in Fig. 3.2.

$$\dot{x}_5 = 0.2333\,(u_1 - q_1(x_5 + 3\dot{x}_5)) \tag{3.37}$$

$$\dot{x}_5(1 + 0.7q_1) = 0.2333u_1 - 0.2333q_1x_5 \tag{3.38}$$

We may eliminate the uncertainty in the $(1+0.7q_1)$ term, since the average value of q_1 is so close to both the upper and lower bound values: $q_{1_{avg}} = 0.9963$, so $(1+0.7q_1) \approx$

1.697. This is a very small approximation and should have no affect on the robustness of the system. This approximation is desirable because avoiding uncertainties in the $\mathbf{B}$ matrix will significantly simplify the process of obtaining the uncertain closed loop system matrix later. This gives us:

$$\dot{x}_5 = 0.1375u_1 - [0.1255, 0.1485]x_5 \tag{3.39}$$

The state equations for the remaining integrator states are calculated in the same fashion.

$$F_{6/150} = q_2(x_6 + \frac{1}{Reset}\dot{x}_6) = [47.44, 49.54](x_6 + 1.5\dot{x}_6) \tag{3.40}$$

$$\begin{aligned} \dot{x}_6 &= Gain \times Reset(u_2 - F_{6/150}/Range) && (3.41)\\ &= 0.5\,(u_2 - [0.9779, 1.021](x_6 + 1.5\dot{x}_6)) && (3.42) \end{aligned}$$

The *Gain*, *Reset*, and *Range* values are shown in Fig. 3.4.

$$\dot{x}_6 = 0.5\,(u_2 - q_2(x_6 + 1.5\dot{x}_6)) \tag{3.43}$$

$$\dot{x}_6(1 + 0.75q_2) = 0.5u_2 - 0.5q_2x_6 \tag{3.44}$$

Again, take the average q_2: $q_{2_{avg}} = 0.9995$, so $(1 + 0.75q_2) \approx 1.750$. This results in:

$$\dot{x}_6 = 0.2857u_2 - [0.2794, 0.2917]x_6 \tag{3.45}$$

$$F_{6/50} = q_3(x_7 + \frac{1}{Reset}\dot{x}_7) = [262.6, 274.4](x_7 + 1.5\dot{x}_7) \tag{3.46}$$

$$\begin{aligned} \dot{x}_7 &= Gain \times Reset(u_3 - F_{6/50}/Range) && (3.47)\\ &= 0.5\,(u_3 - [0.9820, 1.026](x_7 + 1.5\dot{x}_7)) && (3.48) \end{aligned}$$

The *Gain*, *Reset*, and *Range* values are shown in Fig. 3.3.

$$\dot{x}_7 = 0.5\,(u_3 - q_3(x_7 + 1.5\dot{x}_7)) \tag{3.49}$$

$$\dot{x}_7(1 + 0.75)q_3 = 0.5u_3 - 0.5q_3x_7 \tag{3.50}$$

Take: $(1+0.75)q_3) \approx 1.753$.

$$\dot{x}_7 = 0.2852u_3 - [0.2801, 0.2926]x_7 \tag{3.51}$$

$$F_{150/50} = q_4(x_8 + \frac{1}{Reset}\dot{x}_8) = [46.70, 51.39](x_8 + 1.5\dot{x}_8) \tag{3.52}$$

$$\begin{aligned} \dot{x}_8 &= Gain \times Reset(u_4 - F_{150/50}/Range) && (3.53)\\ &= 0.5\,(u_4 - [0.9745, 1.072](x_8 + 1.5\dot{x}_8)) && (3.54) \end{aligned}$$

The *Gain*, *Reset*, and *Range* values are shown in Fig. 3.5.

$$\dot{x}_8 = 0.5\,(u_4 - q_4(x_8 + 1.5\dot{x}_8)) \tag{3.55}$$

$$\dot{x}_8(1+0.75q_4) = 0.5u_4 - 0.5q_4x_8 \tag{3.56}$$

Take: $(1+0.75q_4) \approx 1.767$.

$$\dot{x}_8 = 0.2830u_4 - [0.2757, 0.3033]x_8 \tag{3.57}$$

Next, revisit the header pressure state equations and substitute. For the 900# header:

$$\dot{x}_1 = \frac{-1}{444.1}[244.0, 288.7](x_5 + 3\dot{x}_5) \tag{3.58}$$

$$\dot{x}_1 = -[0.5494, 0.6501](x_5 + 0.4125u_1 - [0.3765, 0.4455]x_5) \tag{3.59}$$

$$\dot{x}_1 = -[0.5494, 0.6501](0.4125u_1 + [0.5545, 0.6235]x_5) \tag{3.60}$$

$$\dot{x}_1 = -[0.5494, 0.6501](0.4125u_1 + [0.5545, 0.6235]x_5) \tag{3.61}$$

Again, we will avoid uncertainties entering into the **B** matrix by taking the average value of $[0.5494, 0.6501] = 0.5998$, and this gives us the usable state equation:

$$\dot{x}_1 = -[0.3046, 0.4053]x_5 - 0.2474u_1 \tag{3.62}$$

For the 600# header:

$$\dot{x}_2 = \frac{1}{1349}\{[244.0, 288.7](x_5 + 3\dot{x}_5) - [47.44, 49.54](x_6 + 1.5x_6) - [262.6, 274.4](x_7 + 1.5\dot{x}_7)\} \tag{3.63}$$

which leads to:

$$\begin{aligned}\dot{x}_2 = & + & [0.1003, 0.1334]x_5 + 0.08145u_1 \\ & - & [0.01978, 0.02133]x_6 - 0.01541u_2 \\ & - & [0.1092, 0.1179]x_7 - 0.08515u_3 \end{aligned} \tag{3.64}$$

For the 150# header:

$$\dot{x}_3 = \frac{1}{1954}\{[47.44, 49.54](x_6 + 1.5\dot{x}_6) - [46.70, 51.39](x_8 + 1.5x_8)\} \tag{3.65}$$

$$\begin{aligned}\dot{x}_3 = & + & [0.01366, 0.01473]x_6 + 0.01064u_2 \\ & - & [0.01302, 0.01542]x_8 - 0.01065u_4 \end{aligned} \tag{3.66}$$

For the 50# header:

$$\dot{x}_4 = \frac{1}{1.530 \times 10^4}\{[262.6, 274.4](x_7 + 1.5\dot{x}_7) + [46.70, 51, 39](x_8 + 1.5x_8)\} \tag{3.67}$$

$$\begin{aligned}\dot{x}_4 = & + & [0.009628, 0.01040]x_7 + 0.007506u_3 \\ & + & [0.001663, 0.001970]x_8 - 0.001361u_4 \end{aligned} \tag{3.68}$$

Putting it all in matrix form and redefining all the q_j's for convenience, the following **A** and **B** matrices were calculated:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & q_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_2 & q_3 & q_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_5 & 0 & q_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_7 & q_8 \\ 0 & 0 & 0 & 0 & q_9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{11} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{12} \end{bmatrix} \tag{3.69}$$

$$\mathbf{B} = \begin{bmatrix} -0.2474 & 0 & 0 & 0 \\ 0.08145 & -0.01541 & -0.08515 & 0 \\ 0 & 0.01064 & 0 & -0.01065 \\ 0 & 0 & 0.007506 & 0.001361 \\ 0.1365 & 0 & 0 & 0 \\ 0 & 0.2857 & 0 & 0 \\ 0 & 0 & 0.2852 & 0 \\ 0 & 0 & 0 & 0.2830 \end{bmatrix} \tag{3.70}$$

where:

$$\begin{aligned}
q_1 &= [-0.4455, -0.3046] \\
q_2 &= [0.1003, 0.1334] \\
q_3 &= [-0.02133, -0.01978] \\
q_4 &= [-0.1179, -0.1092] \\
q_5 &= [0.01366, 0.01473] \\
q_6 &= [-0.01542, -0.01302] \\
q_7 &= [0.009628, 0.01040] \\
q_8 &= [0.001663, 0.001970] \\
q_9 &= [-0.1485, -0.1255] \\
q_{10} &= [-0.2917, -0.2794] \\
q_{11} &= [-1.072, -0.9745] \\
q_{12} &= [-0.3033, -0.2757]
\end{aligned}$$

A quick check of the controllability matrix for the nominal case confirms that this is a controllable system.

Chapter 4

Robust Letdown Controller Design

The complexity of Syncrude's steam letdown system is such that it would be very difficult to successfully propose a control system drastically different from the existing control system. The existing letdown controllers are all cascade type, with a pressure controller giving a setpoint to a flow controller. Therefore, a reasonable compromise is to only replace the pressure controllers. For this study they will be replaced with a MIMO controller which coordinates the letdown flows in a multivariable fashion. Its purpose will be to handle transients on the system more than steady state setpoint achievement since there are other systems in place for that, such as the utility boilers controlling the 900# header pressure and the backpressure turbine generators controlling the 50# header pressure. Both of these are slow to respond to transients, whereas the letdowns are fast, so transient performance improvements shall be expected. Also, since these controllers will only be giving setpoints to the letdowns, they cannot control the steady state 900# header pressure adequately because they cannot make steam, and they cannot control the steady state 50# header pressure adequately because they cannot unmake steam. Two such controllers are presented; the first uses full state feedback and the second uses output feedback.

4.1 MIMO State Feedback Controller

The standard quadratic objective function was selected:

$$J = \int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})dt \tag{4.1}$$

with the following weight parameters:

$$\mathbf{Q} = \mathrm{diag}(1, 1, 1, 0.1, 0, 0, 0, 0) \tag{4.2}$$

$$\mathbf{R} = 0.1 \times \mathbf{I}_4 \tag{4.3}$$

Q and **R** were selected with the assumption that the performance of the 900#, 600#, and 150# headers were more important than the performance of the 50# header and the control effort. The values chosen were settled upon after a small amount of trial and error. MATLAB's `lqr` command was used to calculate the gain matrix for the control law $\mathbf{u} = -\mathbf{K}\mathbf{x}$. Nominal parameters were used in place of the uncertainties. The nominal parameters were taken from a typical plant operating point, and are not simply averages.

$$\mathbf{K} = \left[\begin{array}{cccc} -3.076 & 0.7376 & 0.004877 & 0.002415 \\ -0.1383 & -0.5711 & 2.116 & -0.7194 \\ -0.7187 & -3.021 & -0.3205 & 0.1593 \\ -0.03323 & -0.1017 & -2.328 & -0.6760 \end{array}\right. \cdots$$

$$\cdots \left.\begin{array}{cccc} 1.125 & -0.006732 & -0.03780 & -0.0001118 \\ -0.006292 & 0.1035 & 0.08354 & -0.08676 \\ -0.03575 & 0.08368 & 0.5402 & 0.01460 \\ -9.833 \times 10^{-5} & -0.08706 & 0.01463 & 0.08842 \end{array}\right] \tag{4.4}$$

The closed loop system is described by:

$$\tilde{\mathbf{A}} = (\mathbf{A} - \mathbf{B}\mathbf{K}) \tag{4.5}$$

The four Kharitonov polynomials were calculated numerically by varying the twelve parametric uncertainties, calculating the corresponding closed loop matrix $\tilde{\mathbf{A}}$ and determining the characteristic polynomial. A script file for this "brute force" method, called `charpoly.m`, is shown in Appendix A.7. Note that `charpoly.m` only tests all

combinations of the upper and lower bounds and not values in between, so again not all parameter combinations are accounted for. Also note that we have taken twelve uncertain parameters and represented them in an eighth order interval polynomial, which obviously only has eight uncertain parameters. The resultant characteristic polynomial with an independent uncertainty structure is:

$$\begin{aligned} p(s) = s^8 &+ [2.733, 2.808]s^7 + [3.081, 3.315]s^6 \cdots \\ &+ [1.864, 2.129]s^5 + [0.6511, 0.7978]s^4 \cdots \\ &+ [0.1294, 0.1717]s^3 + [0.01310, 0.01897]s^2 \end{aligned} \tag{4.6}$$

In generating the above interval polynomial, the roots of the uncertain matrix were calculated for every combination of the upper and lower uncertainty bounds of the original twelve parameters, so we already know that the polynomial is robustly stable, but we will use Kharitonov's Theorem anyhow. Fig. 4.2 and Fig. 4.3 show Kharitonov rectangle sweeps for this polynomial. Note that there will always be two poles at the origin, but this is not a concern since there are zeros at the origin to cancel the poles. The four Kharitonov characteristic polynomials are:

$$\begin{aligned} p_1(s) = {} & 0.01897s^2 + 0.1717s^3 + 0.6511s^4 + 1.864s^5 \cdots \\ & +3.315s^6 + 2.808s^7 + s^8 \end{aligned} \tag{4.7}$$

$$\begin{aligned} p_2(s) = {} & 0.01310s^2 + 0.1294s^3 + 0.7978s^4 + 2.129s^5 \cdots \\ & +3.081s^6 + 2.733s^7 + s^8 \end{aligned} \tag{4.8}$$

$$\begin{aligned} p_3(s) = {} & 0.01310s^2 + 0.1717s^3 + 0.7978s^4 + 1.864s^5 \cdots \\ & +3.081s^6 + 2.808s^7 + s^8 \end{aligned} \tag{4.9}$$

$$\begin{aligned} p_4(s) = {} & 0.01897s^2 + 0.1294s^3 + 0.6511s^4 + 2.129s^5 \cdots \\ & +3.315s^6 + 1.864s^7 + s^8 \end{aligned} \tag{4.10}$$

These four polynomials are stable and therefore the polynomial family in Eq. 4.6 is robustly stable, so it would be tempting to conclude that the closed loop system is

robustly stable. However, because not all of the uncertainty was accounted for we cannot claim robust stability. We can, however, claim that because we have accounted for some of the uncertainty, the new design should be more robust than a design that does not account for any uncertainty. Also note that if our resulting polynomial was not robustly stable, then we could select the gain matrix using the state feedback form of the multiple model algorithm, `simstab4sf.m`, in Appendix A.6.

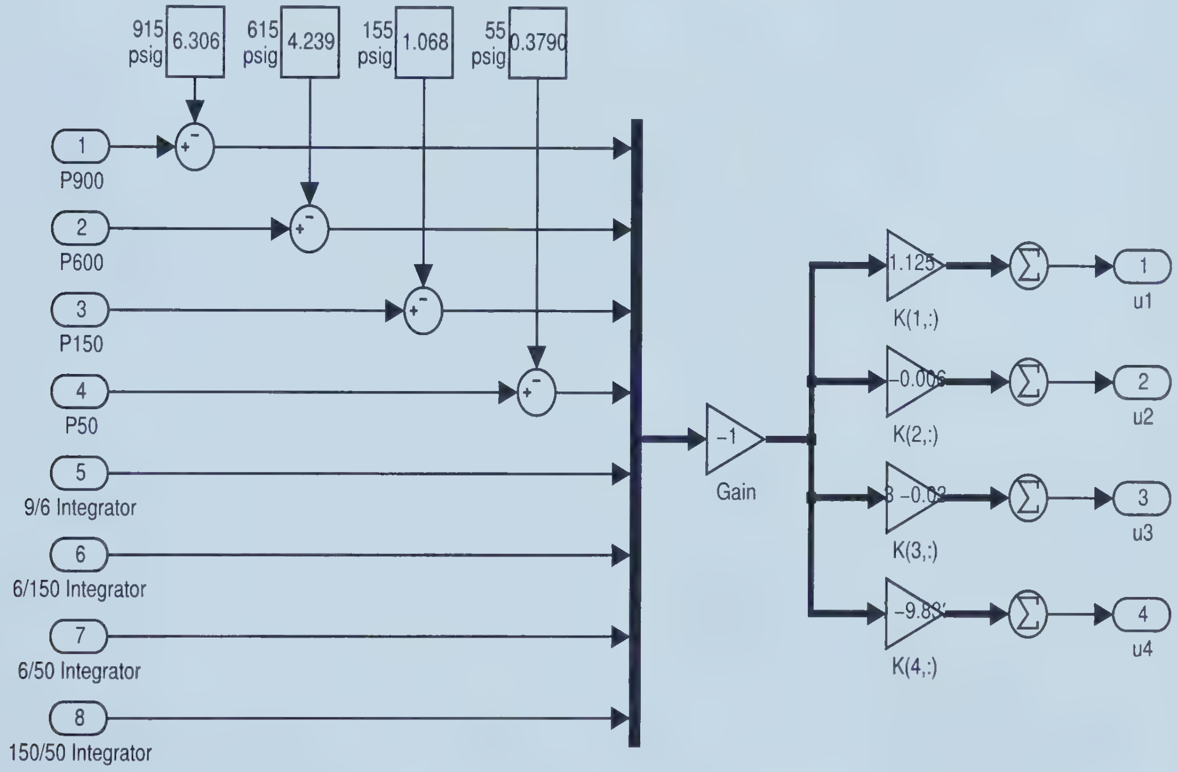

Figure 4.1: Simulink block diagram implementation of the MIMO state feedback controller.

4.2 MIMO Output Feedback Controller

Although we developed a first principles model of the system with carefully selected measurable state and we could simply use state feedback for the controllers, this is often not possible. With this in mind, an output feedback controller is developed using the same first principles model, although in practice it is likely that only system identification models would be available.

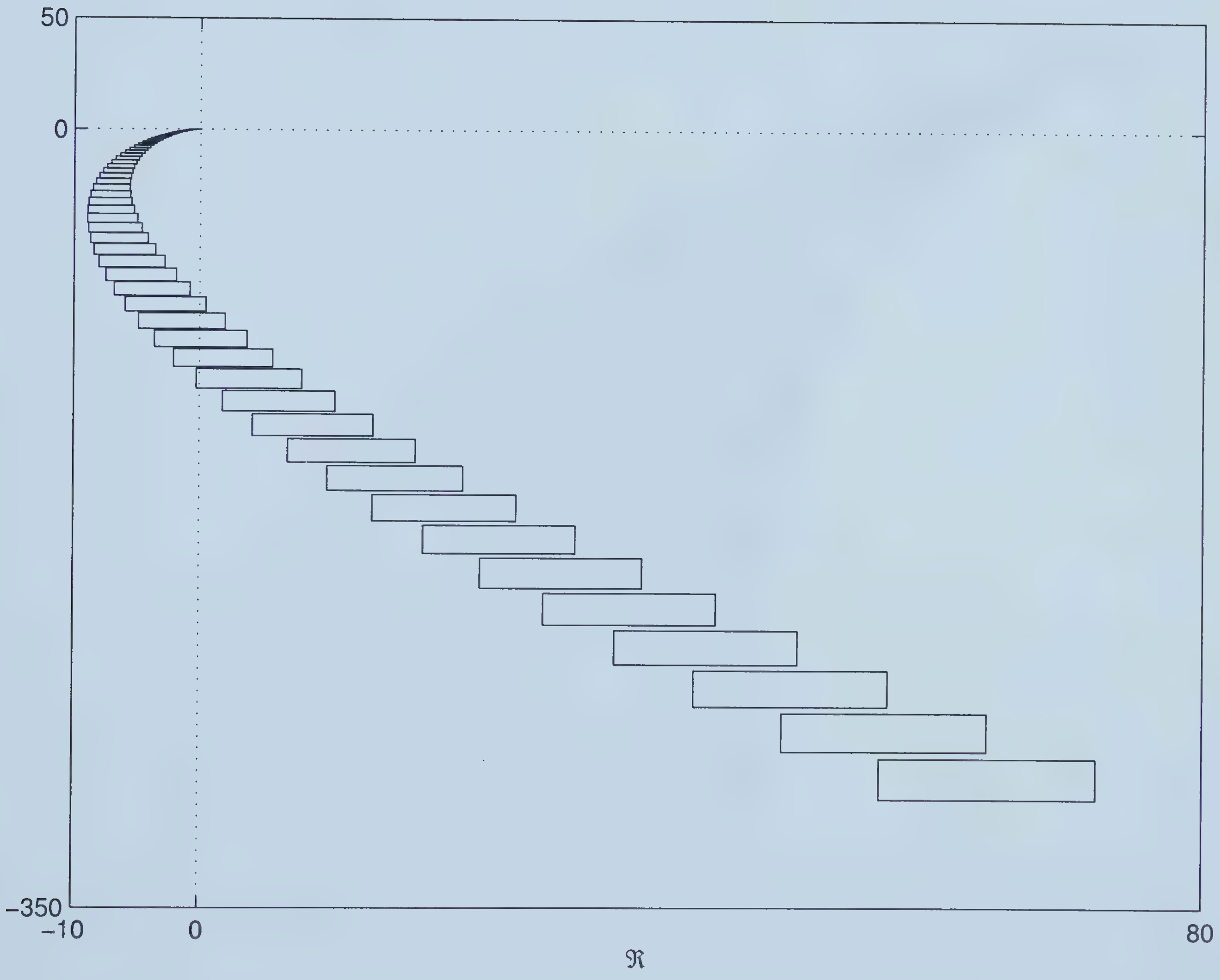

Figure 4.2: Kharitonov rectangle sweep of the uncertain polynomial in Eq. 4.6 for $\omega = 0, \cdots, 2$.

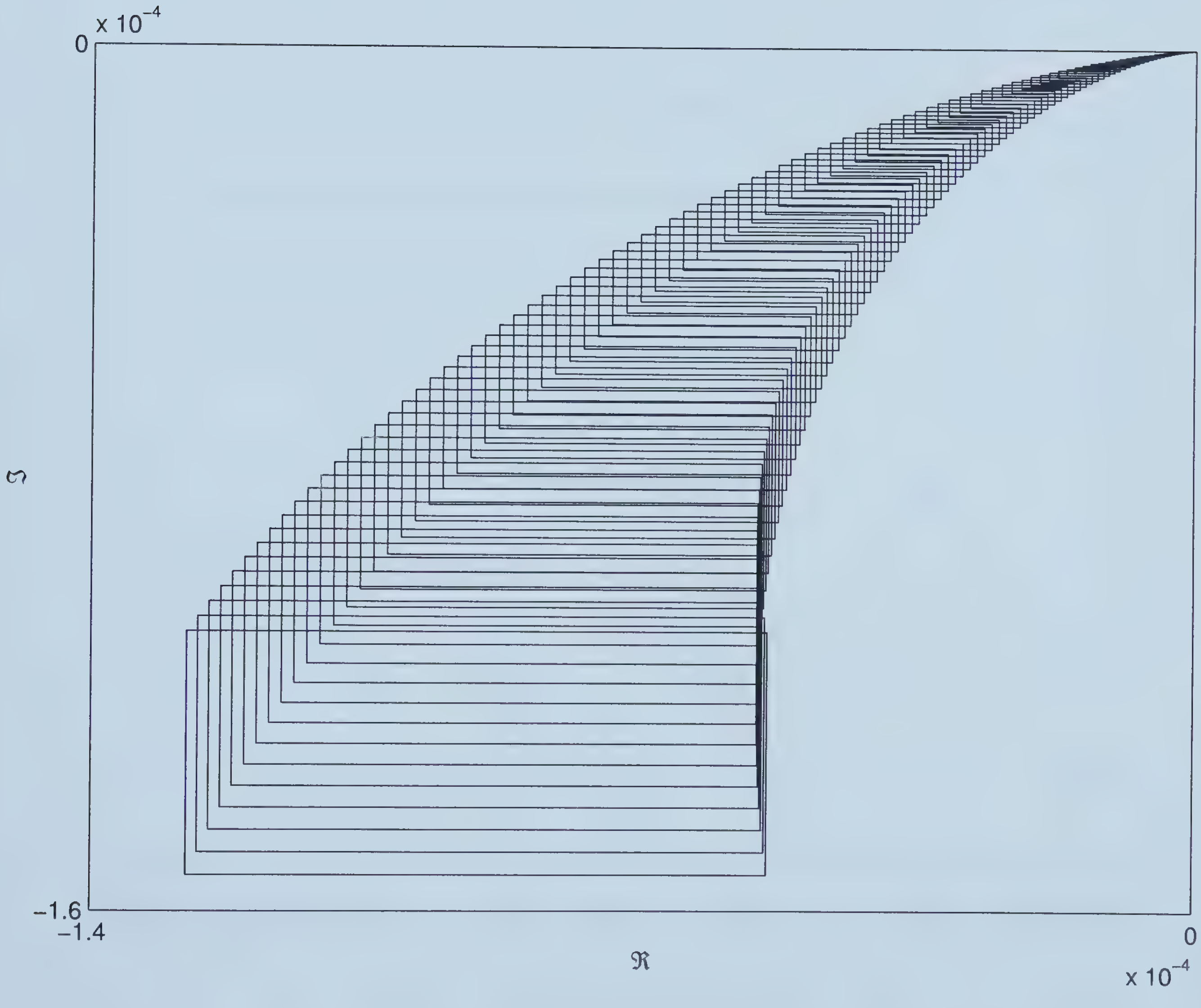

Figure 4.3: Kharitonov rectangle sweep of the uncertain polynomial in Eq. 4.6 for $\omega = 0, \cdots, 0.1$.

The four selected outputs are each of the header pressures with their nominal values subtracted.

$$\begin{aligned} y_1 &= P_{900} - 915\ psig \\ y_2 &= P_{600} - 615\ psig \\ y_3 &= P_{150} - 155\ psig \\ y_4 &= P_{50} - 55\ psig \end{aligned}$$

The control inputs are still the letdown station flow setpoints:

$$\begin{aligned} u_1 &= SP_{900/600\#\ letdown\ flow} \\ u_2 &= SP_{600/150\#\ letdown\ flow} \\ u_3 &= SP_{600/50\#\ letdown\ flow} \\ u_4 &= SP_{150/50\#\ letdown\ flow} \end{aligned}$$

We now require a measurement matrix:

$$\begin{aligned} \mathbf{C} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_4 & \mathbf{0}_{4\times4} \end{bmatrix} \end{aligned} \tag{4.11}$$

Note that we are simply selecting the first four states as outputs. We will also choose the same quadratic objective function weights as with our state feedback controller:

$$\mathbf{Q} = diag(1, 1, 1, 0.1, 0, 0, 0, 0) \tag{4.12}$$

$$\mathbf{R} = 0.1 \times \mathbf{I}_4 \tag{4.13}$$

Now, we use the same "brute force" approach to generate the four Kharitonov plants, but we must modify the algorithm to make it open loop and generate the four Kharitonov **A** matrices instead of the uncertain characteristic polynomial. Note that this method has the same drawbacks of representing twelve uncertain parameters

by eight new uncertain parameters, and not all of the parametric combinations are accounted for. The script file, called `amatrix.m`, is shown in Appendix A.8. The resulting Kharitonov **A** matrices are:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -0.009421 & -0.1526 & -0.9739 & -1.816 \end{bmatrix} \tag{4.14}$$

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -0.01408 & -0.2027 & -0.8099 & -1.655 \end{bmatrix} \tag{4.15}$$

$$\mathbf{A}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -0.01408 & -0.1526 & -0.8099 & -1.8155 \end{bmatrix} \tag{4.16}$$

$$\mathbf{A}_4 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -0.009421 & -0.2027 & -0.9739 & -1.655 \end{bmatrix} \tag{4.17}$$

Running the multiple model algorithm, `simstab4.m`, the algorithm found a locally optimal Kharitonov gain after 162 iterations, however, the algorithm stopped because it reached the limit of its step size (α) tolerance, 1×10^{-6}. Pushing it beyond that point resulted in the quadratic objective function "blowing up" due to poorly scaled

matrices. Therefore, the gain might not be very well optimized, but at least it is guaranteed to be robustly stable for the four Kharitonov matrices, but again not necessarily for the closed loop plant. The gain is:

$$\mathbf{G} = \begin{bmatrix} -1.027 & 16.36 & 11.30 & 23.76 \\ 1.849 & 6.906 & 4.759 & 7.982 \\ -0.8125 & -10.77 & -6.333 & -11.67 \\ 0.7664 & -0.1836 & 0.4086 & -2.001 \end{bmatrix} \tag{4.18}$$

The eigenvalues for each closed loop Kharitonov matrix are:

$$\begin{aligned} \text{eig}(\tilde{\mathbf{A}}_1) &= \{-1.666 \pm 0.6806i, -0.01653 \pm 1.186i, \cdots \\ &\cdots -0.3725 \pm 1.061i, -0.02921 \pm 0.3118i\} && (4.19) \\ \text{eig}(\tilde{\mathbf{A}}_2) &= \{-1.612 \pm 0.6640i, -0.01533 \pm 1.237i, \cdots \\ &\cdots -0.3466 \pm 0.9812i, -0.03057 \pm 0.2778i\} && (4.20) \\ \text{eig}(\tilde{\mathbf{A}}_3) &= \{-1.659 \pm 0.6438i, -0.001191 \pm 1.227i, \cdots \\ &\cdots -0.3728 \pm 0.9810i, -0.05095 \pm 0.1566i\} && (4.21) \\ \text{eig}(\tilde{\mathbf{A}}_4) &= \{-1.619 \pm 0.6955i, -0.04160 \pm 1.188i, \cdots \\ &\cdots -0.3312 \pm 1.070i, -0.01248 \pm 0.3848i\} && (4.22) \end{aligned}$$

The real components of the eigenvalues are all less than zero, which confirms that the four matrices are robustly stable, but not necessarily the closed loop system. However, once again we note that the design does at least account for some of the uncertainty and is an improvement over methods which do not account for uncertainty.

4.3 Comparison of BPSTG Trip Results

A backpressure steam turbogenerator (BPSTG) trip was simulated since this is one of the most harsh plant upsets to the headers, since it involves simultaneously losing a large 900# steam load and a large 50# steam supplier. Naturally, this was simulated on *SynSim* and not actually performed as a live test. Three simulation runs were performed, each with a different controller or set of controllers. The first used the

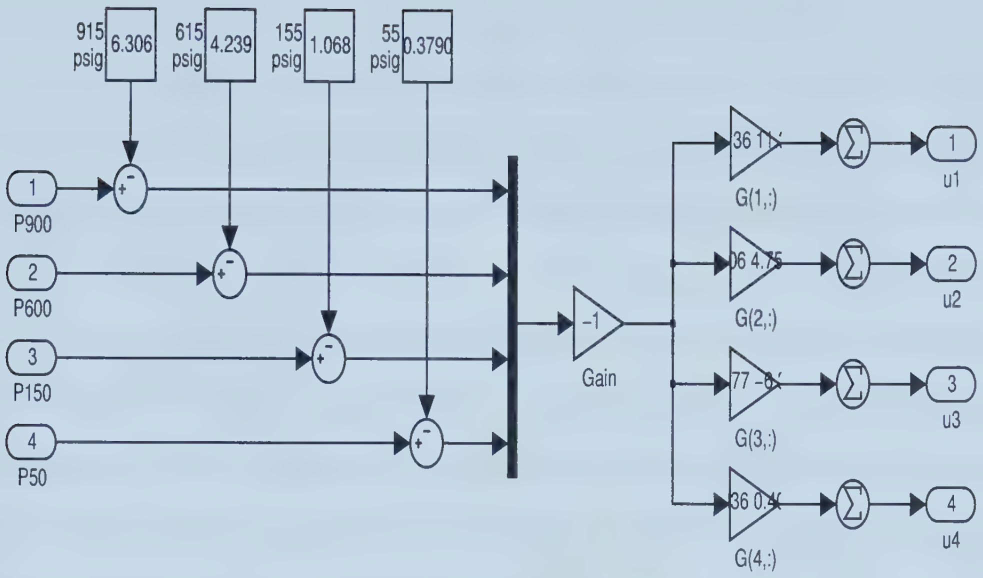

Figure 4.4: Simulink block diagram implementation of the MIMO output feedback controller.

existing cascaded PI controllers, the second used the MIMO state feedback controller, and the last used the MIMO output feedback controller. Numerous graphs from the simulation runs are shown in Appendix B.

Not surprisingly, the state feedback controller performed the best. Comparing the 900# header pressure responses, see Figures B.1, B.5 and B.9, the state feedback controller had the least overshoot, only allowing the header pressure to rise to about 945 psig, while the existing controllers allowed nearly 960 psig and the output feedback controller had 975 psig. Both the state feedback and output feedback controllers returned to setpoint in a very similar fashion with negligible undershoot, whereas the existing controllers allowed the header to drop below 905 psig, where the backpressure turbine generators start to run back as part of the header protection scheme. Also notice the oscillation on the 900# header response with the existing controllers. This is due to the 900/600# letdown and 600/50# letdown fighting, mostly due to the overrides on the 600/50# letdown.

With the 50# header, the state feedback controller was clearly superior again. It allowed the pressure drop to about 35 psig, whereas the existing controllers had

the pressure fall to 30 psig, and the output feedback controller had it fall to 28 psig. The state feedback controller did not return the header pressure to its original setpoint, instead falling into PV25 control, which throttles a large 50# steam load. Remember that both the state and output feedback controllers rely on other systems to achieve steady state header pressure setpoints. Also observe that although the output feedback controller returned the 50# header to its original setpoint, it was initially under PV25 control and not backpressure turbine control as was the case with the state feedback controller, even though both systems had the same initial balances. Again, there is oscillation on the 50# header response with the existing controllers, for the same reasons as stated before.

The real value in the output feedback controller is that it uses far less letdown flow than the other two controllers. This is mostly due to the weights selected in the quadratic objective function. The output feedback controller hardly required any 600/50# flow, and used less than half of the 900/600# flow, which means improved energy efficiency since letdown flow is wasteful. Similar responses may be obtained with the state feedback controller by adjusting its weights to increase the cost of control, and likewise the pressure responses with the output feedback controller can be improved by decreasing the cost of control or by putting greater weight on the output performance. The optimal settings would have to be studied as a tradeoff between energy efficiency and pressure transient performance.

Chapter 5

Conclusions

It has been demonstrated that Kharitonov's Theorem is useful not only for robustness analysis, but also for robust controller synthesis. An overview of Kharitonov's Theorem and some related tools were presented. Simultaneous stabilization techniques were discussed and one algorithm investigated for MIMO state space controllers.

The multiple model algorithm [13] was found to be a useful output feedback controller design tool to account for some, but not all, uncertainty in a closed loop system. It successfully handles the problem of finding an initial stabilizing gain by starting with a set of full state feedback gains and using equality constraints to ensure that an equivalent output feedback gain exists. A small drawback, however, is that it requires the measurement matrix, **C**, to be full rank in order to implement the equality constraints.

5.1 Areas For Further Research

Multiple Model Quadratic Synthesis Gain Scheduling

Broussard and McLean have extended their multiple model algorithm to calculate multiple independent gain scheduling parameters [13]. They use a gain scheduling control law of the form:

$$\mathbf{u}_j = \left(\mathbf{G}_c + q_j \mathbf{G}_q\right) \mathbf{C}_j \mathbf{x}_j \tag{5.1}$$

Where q_j is the gain scheduling parameter which varies with plant condition. The necessary modifications to the algorithm are minor and gain scheduling control designs can readily be generated for study.

Polytopic Model Algorithm

The multiple model algorithm will work for any number of models and not just for the four Kharitonov models or the Kharitonov models plus a nominal model. It is therefore desirable to reduce design conservatism by using a polytopic model derived from a set of generators. However, because of the convexity requirement, even if a stabilizing output feedback gain was found for each generator, there is still no guarantee polytopic system is stable for the computed gain. Therefore, in order to check the stability, it is necessary to simultaneously solve a set of linear matrix inequalities (LMI's) of the following form [35]:

$$\tilde{\mathbf{A}}_j\mathbf{X} + \mathbf{X}\tilde{\mathbf{A}}_j^T < 0 \qquad j = 1, \cdots, N \tag{5.2}$$

$$\mathbf{X} > 0 \tag{5.3}$$

Where the closed loop equation is now in the output feedback form: $\tilde{\mathbf{A}}_j = \mathbf{A}_j - \mathbf{B}_j\mathbf{G}\mathbf{C}_j$. LMI is a relatively recent numerical programming technique and this particular application has already received attention [27] [28] but is still an active area of research [16].

Optimizing Robust Performance Over An Interval Plant Family Using P, PI, or PID Controllers

This is a long standing open question, and research such as that of Ho, Datta, and Bhattacharyya [20] [21] [22] can be extended to investigate the question, and perhaps even to solve it.

Bibliography

[1] Ross B. Barmish. A generalization of Kharitonov's four polynomial concept for robust stability problems with linearly dependent coefficient perturbations. *IEEE Transactions on Automatic Control*, 34(2):157–165, 1989.

[2] Ross B. Barmish. *New Tools for Robustness of Linear Systems*, chapter 1. Macmillan Publishing Company, 1994.

[3] Ross B. Barmish. *New Tools for Robustness of Linear Systems*, chapter 4. Macmillan Publishing Company, 1994.

[4] Ross B. Barmish. *New Tools for Robustness of Linear Systems*, chapter 5. Macmillan Publishing Company, 1994.

[5] Ross B. Barmish. *New Tools for Robustness of Linear Systems*, chapter 10. Macmillan Publishing Company, 1994.

[6] Ross B. Barmish. *New Tools for Robustness of Linear Systems*, chapter 11. Macmillan Publishing Company, 1994.

[7] Bechtel, Inc. *Thermal Electric Study on the Steam Systems and Electrical Generation*, volume 2. Bechtel, Inc., San Francisco, California, 1976. Bechtel Job Number 10997.

[8] Caroll O. Bennet and John Earle Myers. *Momentum, Heat, and Mass Transfer.* Chemical Engineering Series. McGraw-Hill, Inc., 1962.

[9] Dennis S. Bernstein and Wassim M. Haddad. Robust controller synthesis using Kharitonov's Theorem. *IEEE Transactions on Automatic Control*, 37:129–132, 1992.

[10] S.P. Bhattacharyya, H. Chapellat, and L.H. Keel. *Robust Control: The Parametric Approach.* Prentice Hall, 1995.

[11] Vincent Blondel. *Simultaneous Stabilization of Linear Systems*, volume 191 of *Lecture Notes in Control and Information Sciences.* Springer-Verlag, London, 1994.

[12] John R. Broussard. An approach to the optimal output feedback initial stabilizing gain problem. In *Proceedings of the 29th Conference on Decision and Control*, pages 2918–2920, 1990.

[13] John R. Broussard and Chris S. McLean. An algorithm for Kharitonov synthesis. In *Proceedings of the American Control Conference*, pages 1408–1413, 1992.

[14] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control.* Hemisphere Publishing Corporation, 1975.

[15] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. Technical report, Department of Electrical Engineering and Computer Science, Northwestern University, May 1994. Report NAM-08.

[16] Yong-Yan Cao, You-Xian Sun, and James Lam. Simultaneous stabilization via static output feedback and state feedback. *IEEE Transactions on Automatic Control*, 44(6):1277–1282, June 1999.

[17] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison Wesley, 2 edition, 1991.

[18] S.L. Gandhi, J. Graham, M.A. Duffield, and R.M. Cortes. Dynamic simulation analyzes expanded refinery steam system. *Hydrocarbon Processing*, pages 55–65, November 1995.

[19] Didier Henrion, Martin Hromcik, Huibert Kwakernaak, Sonja Pejchová, Michael Sebek, and Rens C. Strijbos. *Polynomial Toolbox Version 1.6*, June 1998.

[20] Ming-Tzu Ho, Aniruddha Datta, and S.P. Bhattacharyya. A New Approach to Feedback Design Part I: Generalized Interlacing and Proportional Control. Technical report, Department of Electrical Engineering, Texas A&M University, College Station, TX, USA, 1997. TAMU-ECE97-001-A.

[21] Ming-Tzu Ho, Aniruddha Datta, and S.P. Bhattacharyya. A New Approach to Feedback Design Part II: PI and PID Controllers. Technical report, Department of Electrical Engineering, Texas A&M University, College Station, TX, USA, 1997. TAMU-ECE97-001-B.

[22] Ming-Tzu Ho, Aniruddha Datta, and S.P. Bhattacharyya. Design of P, PI, and PID controllers for interval plants. In *Proceedings of the American Control Conference*, pages 2496–2501, Philadelphia, Pennsylvania, June 1998.

[23] Thomas F. Irvine and Peter E. Liley. *Steam and Gas Tables With Computer Equations*. Academic Press, Inc., 1984.

[24] V.L. Kharitonov. Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differentsial'nye Uravneniya*, 14:2086–2088, 1978.

[25] Huibert Kwakernaak and Raphael Sivan. *Linear Optimal Control Systems*. Wiley-Interscience, 1972.

[26] R.A Luke, P. Dorato, and C.T. Abdallah. A survey of state-feedback simultateous stabilization techniques. In *Proceedings of the 2nd Automation Congress*, Montpellier, France, 1996.

[27] Robert A. Luke, Peter Dorato, and Chaouki T. Abdallah. Linear-quadratic simultaneous performance design. In *Proceedings of the American Controls Conference*, Albuquerque, New Mexico, 1997.

[28] Robert Anthony Luke. *Multiple Model State Feedback Performance Design Using Linear Matrix Inequalities*. PhD thesis, University of New Mexico, December 1997.

[29] Jan Lunze. *Robust Multivariable Feedback Control*. Prentice Hall International Series In Systems and Control Engineering. Prentice Hall, 1989.

[30] Pertti Makila and Hannu Toivonen. Computational methods for parametric LQ problems - a survey. *IEEE Transactions on Automatic Control*, 32:658-671, 1987.

[31] Daniel D. Moerder and Anthony J. Calise. Convergence of a numerical algorithm for calculating optimal output feedback gains. *IEEE Transactions on Automatic Control*, 30(9):900–903, September 1985.

[32] Celso J. Munaro and Celso P. Bottura. On the simultaneous stabilization of dynamic systems via linear control. In *Proceedings of the 13th Triennial World Congress*, volume G, pages 465–468, San Francisco, USA, 1996. Internation Federation of Automatic Control.

[33] Rajeev Kumar Ranjan. Parametric approach to steady-state stability analysis of power systems. Master's thesis, University of Illinois at Urbana-Champaign, 1992.

[34] W.E. Schmitendorf and C.V. Hollot. Simultaneous stabilization via linear state feedback control. *IEEE Transactions on Automatic Control*, 34(9):1001–1005, September 1989.

[35] Andreas Varga. Optimal output feedback control: A multi-model approach. Technical report, German Aerospace Research Establishment Institute for Robotics and System Dynamics, P.O.B. 1116, D-82230 Wessling, Germany, 1999.

[36] Kehui Wei and B. Ross Barmish. An iterative design procedure for simultaneous stabilization of mimo systems. *Automatica*, 24(5):643–652, 1988.

[37] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control.* Prentice Hall, 1996.

[38] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. L-BFGS-B - Fortran subroutines for large-scale bound constrained optimization. Technical report, Department of Electrical Engineering and Computer Science, Northwestern University, December 1994. Report NAM-11.

Appendix A

MATLAB Code

A.1 kpoly

```
function [p1,p2,p3,p4]=kpoly(p)
% function [p1,p2,p3,p4]=kpoly(p)
%
% Generates the four Kharitonov polynomials.
% The interval polynomial p is in two rows, with
% one set of bounds on each row, in descending
% powers of s.  It does not matter which row
% the upper and lower bounds are in.
%
% Author:  David White

order=max(size(p))-1;          % order of polynomial
p1=[];
p2=[];
p3=[];
p4=[];
p=fliplr(p);                   % ascending powers of s
for k=1:order+1
   m=rem(k,4);
   switch m
      case 1
         p1=[p1 min(p(:,k))];
         p2=[p2 max(p(:,k))];
         p3=[p3 max(p(:,k))];
         p4=[p4 min(p(:,k))];
      case 2
         p1=[p1 min(p(:,k))];
         p2=[p2 max(p(:,k))];
         p3=[p3 min(p(:,k))];
         p4=[p4 max(p(:,k))];
      case 3
         p1=[p1 max(p(:,k))];
         p2=[p2 min(p(:,k))];
         p3=[p3 min(p(:,k))];
         p4=[p4 max(p(:,k))];
      case 0
```

```
         p1=[p1 max(p(:,k))];
         p2=[p2 min(p(:,k))];
         p3=[p3 max(p(:,k))];
         p4=[p4 min(p(:,k))];
   end
end
p1=fliplr(p1);                % descending powers of s
p2=fliplr(p2);
p3=fliplr(p3);
p4=fliplr(p4);
```

A.2 cutoff.m

```
function wc=cutoff(p)
% function wc=cutoff(p)
%
% The interval polynomial p is in two rows, with
% one set of bounds on each row, in descending
% powers of s.  It does not matter which row
% the upper and lower bounds are in.
%
% Author:  David White
%
% Reference:  B. Ross Barmish,
%             New Tools For Robustness of Linear Systems
%             MacMillan Publishing Company, 1994
%             Page 80

sizep=size(p);
n=sizep(2);
% First, put the upper bounds in row 1
% and the lower bounds in row 2.
for k=1:n
   if p(2,k)>p(1,k)
      p(:,k)=flipud(p(:,k));
   end
end
% The cutoff frequency is the largest real root of f.
f=[p(2,1) -p(1,2:n)];
wc=max(real(roots(f)));
```

A.3 krsweep.m

```
function krsweep(p,w1,w2,n)
% function krsweep(p,w1,w2,n)
%
% Plots the Kharitonov rectangles for n points in the frequency
% range w1:w2.  The interval polynomial p is in two rows, with
% one set of bounds on each row, in descending powers of s.
% It does not matter which row the upper and lower bounds are in.
% This function requires the function kpoly.m to generate
% the four Kharitonov polynomials.
```

```
%
% Author:  David White

order=max(size(p))-1;         % order of polynomial
[p1,p2,p3,p4]=kpoly(p);       % Kharitonov polynomials
% Plot Kharitonov rectangles.
for w=w1:(w2-w1)/(n-1):w2
   K1=polyval(p1,w*i);
   K2=polyval(p2,w*i);
   K3=polyval(p3,w*i);
   K4=polyval(p4,w*i);
   plot([real(K1) real(K3) real(K2) real(K4) real(K1)], ...
        [imag(K1) imag(K3) imag(K2) imag(K4) imag(K1)])
   hold on
end
hold off
% Put grid markers through origin only.
handle=gca;                % get current figure handle
xlim=get(handle,'XLim');   % x axis limits
ylim=get(handle,'YLim');   % y axis limits
% Ensure monotonically increasing axis limits.
if 0<=xlim(1)|0>=xlim(2)
   xtick=[xlim(1) xlim(2)];
else
   xtick=[xlim(1) 0 xlim(2)];
end
if 0<=ylim(1)|0>=ylim(2)
   ytick=[ylim(1) ylim(2)];
else
   ytick=[ylim(1) 0 ylim(2)];
end
set(handle,'XTick',xtick,'YTick',ytick);
grid on
xlabel('\Re');
ylabel('\Im');
```

A.4 simstab4.m

```
% simstab4.m
%
% Multiple plant model algorithm using the four Kharitonov plants.
%    It is assumed that you have the general state space form:
%       dx/dt=Ax+Bu
%       y=Cx
%    Note:  It is assumed without loss of generality that D=0.
%
%    The following variables are required in the workspace:
%       A1, A2, A3, A4 - Kharitonov matrices
%       B, C - input and output matrices
%       Q, R - performance and control indices
%
% Author:  David White
%
% Reference:  J.R. Broussard and C.S. McLean,
```

```
%              "An Algorithm For Kharitonov Synthesis",
%              Proceedings of the American Control Conference,
%              pp. 1408-1413, 1992.

%Preamble
Rbar=C'*inv(C*C');                         % right inverse of output matrix
nm=4;                                      % Number of plants.
Xt=eye(max(size(A1)));                     % Expected values.
Cbar=C'*inv(C*C')*C;
sizeB=size(B);
m=sizeB(2);                                % Number of controls.
sizeC=size(C);
l=sizeC(1);                                % Number of measurements.
Vbar=null(C);
sizeVbar=size(Vbar);
k=sizeVbar(2);
v=[];
r=[];
for x=1:k
   for y=1:m
      v=[v Vbar(:,x)];
   end
   r=[r;eye(m)];
end
p=0;                                       % Number of additional constraints.
N=m*(k+p);


% Step 0.  Start from any set of full state feedback
%          gains which stabilize the plants.
K1=lqr(A1,B,Q,R);
K2=lqr(A2,B,Q,R);
K3=lqr(A3,B,Q,R);
K4=lqr(A4,B,Q,R);
k=0;
state=1;
alpha=1;
Jn=1;
Jk=2;

while (state==1)|((state==2)&(Jn<=Jk))
% Step 1.  Solve the necessary conditions.
   transition=0;
   A1cl=A1-B*K1;
   A2cl=A2-B*K2;
   A3cl=A3-B*K3;
   A4cl=A4-B*K4;
   P1=lyap(A1cl',Q+K1'*R*K1);
   P2=lyap(A2cl',Q+K2'*R*K2);
   P3=lyap(A3cl',Q+K3'*R*K3);
   P4=lyap(A4cl',Q+K4'*R*K4);
   S1=lyap(A1cl,Xt);
   S2=lyap(A2cl,Xt);
   S3=lyap(A3cl,Xt);
   S4=lyap(A4cl,Xt);
```

```
% Step 2.  Apply the equality constraints.
   X2=B'*P2*S2-R*K2*S2;
   X3=B'*P3*S3-R*K3*S3;
   X4=B'*P4*S4-R*K4*S4;
   for x=1:N
      Z(1,x)=r(x,:)*inv(R)*(B'*P1*S1+X2*Cbar'+X3*Cbar'+ ...
         X4*Cbar')*inv(S1)*v(:,x);
      for y=1:N
         Y(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S1)*v(:,y);
      end
   end
   L=Z*inv(Y);                                 % Lagrange multipliers.

% Step 3.  Compute the gradient for model 1.
   summation=0;
   for x=1:N;
      summation=summation+r(x,:)'*L(N)*v(:,x)';
   end
   K1bar=inv(R)*(B'*P1*S1+X2*Cbar'+X3*Cbar'+X4*Cbar' ...
      -summation)*inv(S1);
   delK1=K1bar-K1;

% Step 4.  If in state 1, compute gradient for remaining models.
   if state==1
      delK2=K1bar*Cbar-K2;
      delK3=K1bar*Cbar-K3;
      delK4=K1bar*Cbar-K4;
      alpha=1;
      eigstate=0;
      while eigstate==0
         K1n=K1+alpha*delK1;
         K2n=K2+alpha*delK2;
         K3n=K3+alpha*delK3;
         K4n=K4+alpha*delK4;
         eig1=eig(A1-B*K1n);
         eig2=eig(A2-B*K2n);
         eig3=eig(A3-B*K3n);
         eig4=eig(A4-B*K4n);
         temp=1;
         for x=1:max(size(eig1))
            if real(eig1(x))>0
               temp=0;
            elseif real(eig2(x))>0
               temp=0;
            elseif real(eig3(x))>0
               temp=0;
            elseif real(eig4(x))>0
               temp=0;
            end
         end
         if temp==0
            alpha=alpha*0.1;
         elseif temp==1;
            eig1a=eig(A1-B*K1n*Rbar*C);
```

```
            eig2a=eig(A2-B*K1n*Rbar*C);
            eig3a=eig(A3-B*K1n*Rbar*C);
            eig4a=eig(A4-B*K1n*Rbar*C);
            temp1=1;
            for x=1:max(size(eig1))
               if real(eig1a(x))>0
                  temp1=0;
               elseif real(eig2a(x))>0
                  temp1=0;
               elseif real(eig3a(x))>0
                  temp1=0;
               elseif real(eig4a(x))>0
                  temp1=0;
               end
            end
            if alpha==1|temp1==1
               state=2;
               fprintf('Initial Kharitonov gain found after %d iterations.\n
               Kinit=K1n;
               Ginit=K1n*Rbar
               transition=1;
            end
            eigstate=1;
         end
      end
      K1=K1n;
      K2=K2n;
      K3=K3n;
      K4=K4n;

% Step 5.  If in state 2, minimize cost function.
   elseif (state==2)&(transition~=1)
      transition=0;
      Jint=trace(P1*Xt)+trace(P2*Xt)+trace(P3*Xt)+trace(P4*Xt);
      J1=0;
      for x=1:N
         J1=J1+r(x,:)*K1*v(:,x)*L(x);
      end
      J2=trace((K2-K1*Cbar)*X2');
      J3=trace((K3-K1*Cbar)*X3');
      J4=trace((K4-K1*Cbar)*X4');
      J(k)=Jint+J1+J2+J3+J4;
      alpha=1;
      Jn=J(k)+1;
      while (Jn>J(k))&(alpha>1e-6)
         K1n=K1+alpha*delK1;
         K2n=K1n*Cbar;
         K3n=K1n*Cbar;
         K4n=K1n*Cbar;
         A1cln=A1-B*K1n;
         A2cln=A2-B*K2n;
         A3cln=A3-B*K3n;
         A4cln=A4-B*K4n;
         P1n=lyap(A1cln',Q+K1n'*R*K1n);
         P2n=lyap(A2cln',Q+K2n'*R*K2n);
```

```
            P3n=lyap(A3cln',Q+K3n'*R*K3n);
            P4n=lyap(A4cln',Q+K4n'*R*K4n);
            S1n=lyap(A1cln,Xt);
            S2n=lyap(A2cln,Xt);
            S3n=lyap(A3cln,Xt);
            S4n=lyap(A4cln,Xt);
            X2n=B'*P2n*S2n-R*K2n*S2n;
            X3n=B'*P3n*S3n-R*K3n*S3n;
            X4n=B'*P4n*S4n-R*K4n*S4n;
            for x=1:N
               Zn(1,x)=r(x,:)*inv(R)*(B'*P1n*S1n+X2n*Cbar'+X3n*Cbar' ...
                  +X4n*Cbar')*inv(S1n)*v(:,x);
               for y=1:N
                  Yn(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S1n)*v(:,y);
               end
            end
            Ln=Zn*inv(Yn);
            Jint=trace(P1n*Xt)+trace(P2n*Xt)+trace(P3n*Xt)+trace(P4n*Xt);
            J1=0;
            for x=1:N
               J1=J1+r(x,:)*K1n*v(:,x)*Ln(x);
            end
            J2=trace((K2n-K1n*Cbar)*X2n');
            J3=trace((K3n-K1n*Cbar)*X3n');
            J4=trace((K4n-K1n*Cbar)*X4n');
            Jn=Jint+J1+J2+J3+J4;
            if Jn>J(k)
               alpha=alpha*0.1;
            end
         end
         if alpha>1e-6
            K1=K1n;
            K2=K2n;
            K3=K3n;
            K4=K4n;
         else
            fprintf('J is minimized within alpha tolerance.\n')
         end
         if k~=0
            Jk=J(k);
         else
            Jn=Jk-1;
         end
      end
      k=k+1;
      if floor(k/100)==(k/100)
         fprintf('%d iterations.  Alpha=%g \n',k,alpha)
      end
   end
   G=K1*Rbar;
   eig1=eig(A1-B*G*C);
   eig2=eig(A2-B*G*C);
   eig3=eig(A3-B*G*C);
   eig4=eig(A4-B*G*C);
   temp=1;
```

```
for x=1:max(size(eig1))
   if real(eig1(x))>0
      temp=0;
   elseif real(eig2(x))>0
      temp=0;
   elseif real(eig3(x))>0
      temp=0;
   elseif real(eig4(x))>0
      temp=0;
   end
end
fprintf('Total iterations:  %d\n',k-1)
if temp==0;
   fprintf('Algorithm failed -- system not robustly stable.\n')
else
   fprintf('Robustly stable optimal output feedback gain is:\n')
   G
end
```

A.5 simstab5.m

```
% simstab5.m
%
% Multiple plant model algorithm using the four Kharitonov plants
% plus a nominal plant to add weight to the optimization of a
% favoured operating condition.
%    It is assumed that you have the general state space form:
%       dx/dt=Ax+Bu
%       y=Cx
%    Note:  It is assumed without loss of generality that D=0.
%
%    The following variables are required in the workspace:
%       A1, A2, A3, A4 - Kharitonov matrices
%       A5 - nominal plant
%       B, C - input and output matrices
%       Q, R - performance and control indices
%
% Author:  David White
%
% Reference:  J.R. Broussard and C.S. McLean,
%             "An Algorithm For Kharitonov Synthesis",
%             Proceedings of the American Control Conference,
%             pp. 1408-1413, 1992.

%Preamble
Rbar=C'*inv(C*C');                       % right inverse of output matrix
nm=5;                                    % Number of plants.
Xt=eye(max(size(A1)));                   % Expected values.
Cbar=C'*inv(C*C')*C;
sizeB=size(B);
m=sizeB(2);                              % Number of controls.
sizeC=size(C);
l=sizeC(1);                              % Number of measurements.
Vbar=null(C);
```

```
sizeVbar=size(Vbar);
k=sizeVbar(2);
v=[];
r=[];
for x=1:k
   for y=1:m
      v=[v Vbar(:,x)];
   end
   r=[r;eye(m)];
end
p=0;                                     % Number of additional constraints.
N=m*(k+p);


% Step 0.  Start from any set of full state feedback
%          gains which stabilize the plants.
K1=lqr(A1,B,Q,R);
K2=lqr(A2,B,Q,R);
K3=lqr(A3,B,Q,R);
K4=lqr(A4,B,Q,R);
K5=lqr(A5,B,Q,R);
k=0;
state=1;
alpha=1;
Jn=1;
Jk=2;

while (state==1)|((state==2)&(Jn<Jk))
% Step 1.  Solve the necessary conditions.
   transition=0;
   A1cl=A1-B*K1;
   A2cl=A2-B*K2;
   A3cl=A3-B*K3;
   A4cl=A4-B*K4;
   A5cl=A5-B*K5;
   P1=lyap(A1cl',Q+K1'*R*K1);
   P2=lyap(A2cl',Q+K2'*R*K2);
   P3=lyap(A3cl',Q+K3'*R*K3);
   P4=lyap(A4cl',Q+K4'*R*K4);
   P5=lyap(A5cl',Q+K5'*R*K5);
   S1=lyap(A1cl,Xt);
   S2=lyap(A2cl,Xt);
   S3=lyap(A3cl,Xt);
   S4=lyap(A4cl,Xt);
   S5=lyap(A5cl,Xt);

% Step 2.  Apply the equality constraints.
   X1=B'*P1*S1-R*K1*S1;
   X2=B'*P2*S2-R*K2*S2;
   X3=B'*P3*S3-R*K3*S3;
   X4=B'*P4*S4-R*K4*S4;
   for x=1:N
      Z(1,x)=r(x,:)*inv(R)*(B'*P5*S5+X1*Cbar'+X2*Cbar'+X3*Cbar'+X4*Cbar')*...
         inv(S5)*v(:,x);
      for y=1:N
```

```
         Y(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S5)*v(:,y);
      end
   end
   L=Z*inv(Y);                                % Lagrange multipliers.

% Step 3.  Compute the gradient for model 5.
   summation=0;
   for x=1:N;
      summation=summation+r(x,:)'*L(N)*v(:,x)';
   end
   K5bar=inv(R)*(B'*P5*S5+X1*Cbar'+X2*Cbar'+X3*Cbar'+X4*Cbar'-summation)*...
      inv(S5);
   delK5=K5bar-K5;

% Step 4.  If in state 1, compute gradient for remaining models.
   if state==1
      delK1=K5bar*Cbar-K1;
      delK2=K5bar*Cbar-K2;
      delK3=K5bar*Cbar-K3;
      delK4=K5bar*Cbar-K4;
      alpha=1;
      eigstate=0;
      while eigstate==0
         K1n=K1+alpha*delK1;
         K2n=K2+alpha*delK2;
         K3n=K3+alpha*delK3;
         K4n=K4+alpha*delK4;
         K5n=K5+alpha*delK5;
         eig1=eig(A1-B*K1);
         eig2=eig(A2-B*K2);
         eig3=eig(A3-B*K3);
         eig4=eig(A4-B*K4);
         eig5=eig(A5-B*K5);
         temp=1;
         for x=1:max(size(eig1))
            if real(eig1(x))>0
               temp=0;
            elseif real(eig2(x))>0
               temp=0;
            elseif real(eig3(x))>0
               temp=0;
            elseif real(eig4(x))>0
               temp=0;
            elseif real(eig5(x))>0
               temp=0;
            end
         end
         if temp==0
            alpha=alpha*0.1;
         elseif temp==1;
            eig1a=eig(A1-B*K1n*Rbar*C);
            eig2a=eig(A2-B*K2n*Rbar*C);
            eig3a=eig(A3-B*K3n*Rbar*C);
            eig4a=eig(A4-B*K4n*Rbar*C);
            eig5a=eig(A5-B*K5n*Rbar*C);
```

```
            temp1=1;
            for x=1:max(size(eig1))
               if real(eig1a(x))>0
                  temp=0;
               elseif real(eig2a(x))>0
                  temp=0;
               elseif real(eig3a(x))>0
                  temp=0;
               elseif real(eig4a(x))>0
                  temp=0;
               elseif real(eig5a(x))>0
                  temp=0;
               end
            end
            if alpha==1|temp1==1;
               state=2;
               fprintf('Initial Kharitonov gain found after %d iterations.\n
               Kinit=K1n;
               Ginit=K1n*Rbar
               transition=1;
            end
            eigstate=1;
         end
      end
      K1=K1n;
      K2=K2n;
      K3=K3n;
      K4=K4n;
      K5=K5n;

% Step 5.  If in state 2, minimize cost function.
   elseif (state==2)&(transition~=1)
      transition=0;
      Jint=0.001*(trace(P1*Xt)+trace(P2*Xt)+trace(P3*Xt)+trace(P4*Xt))+...
         trace(P5*Xt);
      J5=0;
      for x=1:N
         J5=J5+r(x,:)*K5*v(:,x)*L(x);
      end
      J1=trace((K1-K5*Cbar)*X1');
      J2=trace((K2-K5*Cbar)*X2');
      J3=trace((K3-K5*Cbar)*X3');
      J4=trace((K4-K5*Cbar)*X4');
      J(k)=Jint+J1+J2+J3+J4+J5;
      alpha=1;
      Jn=J(k)+1;
      while (Jn>J(k))&(alpha>1e-6)
         K5n=K5+alpha*delK5;
         K1n=K5n*Cbar;
         K2n=K5n*Cbar;
         K3n=K5n*Cbar;
         K4n=K5n*Cbar;
         A1cln=A1-B*K1n;
         A2cln=A2-B*K2n;
         A3cln=A3-B*K3n;
```

```
    A4cln=A4-B*K4n;
    A5cln=A5-B*K5n;
    P1n=lyap(A1cln',Q+K1n'*R*K1n);
    P2n=lyap(A2cln',Q+K2n'*R*K2n);
    P3n=lyap(A3cln',Q+K3n'*R*K3n);
    P4n=lyap(A4cln',Q+K4n'*R*K4n);
    P5n=lyap(A5cln',Q+K5n'*R*K5n);
    S1n=lyap(A1cln,Xt);
    S2n=lyap(A2cln,Xt);
    S3n=lyap(A3cln,Xt);
    S4n=lyap(A4cln,Xt);
    S5n=lyap(A5cln,Xt);
    X1n=B'*P1n*S1n-R*K1n*S1n;
    X2n=B'*P2n*S2n-R*K2n*S2n;
    X3n=B'*P3n*S3n-R*K3n*S3n;
    X4n=B'*P4n*S4n-R*K4n*S4n;
    for x=1:N
      Zn(1,x)=r(x,:)*inv(R)*(B'*P5n*S5n+X1n*Cbar'+X2n*Cbar'+X3n*Cbar'
        +X4n*Cbar')*inv(S5n)*v(:,x);
      for y=1:N
        Yn(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S5n)*v(:,y);
      end
    end
    Ln=Zn*inv(Yn);
    Jint=0.001*(trace(P1n*Xt)+trace(P2n*Xt)+trace(P3n*Xt)+trace(P4n*Xt)
      +trace(P5n*Xt);
    J5=0;
    for x=1:N
      J5=J5+r(x,:)*K5n*v(:,x)*Ln(x);
    end
    J1=trace((K1n-K5n*Cbar)*X1n');
    J2=trace((K2n-K5n*Cbar)*X2n');
    J3=trace((K3n-K5n*Cbar)*X3n');
    J4=trace((K4n-K5n*Cbar)*X4n');
    Jn=Jint+J1+J2+J3+J4+J5;
    if Jn>J(k)
      alpha=alpha*0.1;
    end
  end
  if alpha>1e-6
    K1=K1n;
    K2=K2n;
    K3=K3n;
    K4=K4n;
    K5=K5n;
  else
    fprintf('J is locally minimized within alpha tolerance.\n')
  end
  if k~=0
    Jk=J(k);
  else
    Jn=Jk-1;
  end
  end
k=k+1;
```

```
   if floor(k/100)==(k/100)
      k;
   end
end
G=K5*Rbar;
eig1=eig(A1-B*G*C);
eig2=eig(A2-B*G*C);
eig3=eig(A3-B*G*C);
eig4=eig(A4-B*G*C);
eig5=eig(A5-B*G*C);
temp=1;
for x=1:max(size(eig1))
   if real(eig1(x))>0
      temp=0;
   elseif real(eig2(x))>0
      temp=0;
   elseif real(eig3(x))>0
      temp=0;
   elseif real(eig4(x))>0
      temp=0;
   elseif real(eig5(x))>0
      temp=0;
   end
end
fprintf('Total iterations:  %d\n',k-1)
if temp==0;
   fprintf('Algorithm failed -- system not robustly stable.\n')
else
   fprintf('Robustly stable optimal output feedback gain is:\n')
   G
end
```

A.6 simstab4sf.m

```
% simstab4.m
%
% Multiple plant model algorithm using the four Kharitonov plants.
%    It is assumed that you have the general state space form:
%       dx/dt=Ax+Bu
%       y=Cx
%    Note:  It is assumed without loss of generality that D=0.
%
%    The following variables are required in the workspace:
%       A1, A2, A3, A4 - Kharitonov matrices
%       B, C - input and output matrices
%       Q, R - performance and control indices
%
% Author:  David White
%
% Reference:  J.R. Broussard and C.S. McLean,
%             "An Algorithm For Kharitonov Synthesis",
%             Proceedings of the American Control Conference,
%             pp. 1408-1413, 1992.
```

```
%Preamble
Rbar=C'*inv(C*C');                       % right inverse of output matrix
nm=4;                                    % Number of plants.
Xt=eye(max(size(A1)));                   % Expected values.
Cbar=C'*inv(C*C')*C;
sizeB=size(B);
m=sizeB(2);                              % Number of controls.
sizeC=size(C);
l=sizeC(1);                              % Number of measurements.
Vbar=null(C);
sizeVbar=size(Vbar);
k=sizeVbar(2);
v=[];
r=[];
for x=1:k
   for y=1:m
      v=[v Vbar(:,x)];
   end
   r=[r;eye(m)];
end
p=0;                                     % Number of additional constraints.
N=m*(k+p);


% Step 0.  Start from any set of full state feedback
%          gains which stabilize the plants.
K1=lqr(A1,B,Q,R);
K2=lqr(A2,B,Q,R);
K3=lqr(A3,B,Q,R);
K4=lqr(A4,B,Q,R);
k=0;
state=1;
alpha=1;
Jn=1;
Jk=2;

while (state==1)|((state==2)&(Jn<=Jk))
% Step 1.  Solve the necessary conditions.
   transition=0;
   A1cl=A1-B*K1;
   A2cl=A2-B*K2;
   A3cl=A3-B*K3;
   A4cl=A4-B*K4;
   P1=lyap(A1cl',Q+K1'*R*K1);
   P2=lyap(A2cl',Q+K2'*R*K2);
   P3=lyap(A3cl',Q+K3'*R*K3);
   P4=lyap(A4cl',Q+K4'*R*K4);
   S1=lyap(A1cl,Xt);
   S2=lyap(A2cl,Xt);
   S3=lyap(A3cl,Xt);
   S4=lyap(A4cl,Xt);

% Step 2.  Apply the equality constraints.
   X2=B'*P2*S2-R*K2*S2;
   X3=B'*P3*S3-R*K3*S3;
```

```
   X4=B'*P4*S4-R*K4*S4;
   for x=1:N
      Z(1,x)=r(x,:)*inv(R)*(B'*P1*S1+X2*Cbar'+X3*Cbar'+ ...
         X4*Cbar')*inv(S1)*v(:,x);
      for y=1:N
         Y(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S1)*v(:,y);
      end
   end
   L=Z*inv(Y);                                % Lagrange multipliers.

% Step 3.  Compute the gradient for model 1.
   summation=0;
   for x=1:N;
      summation=summation+r(x,:)'*L(N)*v(:,x)';
   end
   K1bar=inv(R)*(B'*P1*S1+X2*Cbar'+X3*Cbar'+X4*Cbar' ...
      -summation)*inv(S1);
   delK1=K1bar-K1;

% Step 4.  If in state 1, compute gradient for remaining models.
   if state==1
      delK2=K1bar*Cbar-K2;
      delK3=K1bar*Cbar-K3;
      delK4=K1bar*Cbar-K4;
      alpha=1;
      eigstate=0;
      while eigstate==0
         K1n=K1+alpha*delK1;
         K2n=K2+alpha*delK2;
         K3n=K3+alpha*delK3;
         K4n=K4+alpha*delK4;
         eig1=eig(A1-B*K1n);
         eig2=eig(A2-B*K2n);
         eig3=eig(A3-B*K3n);
         eig4=eig(A4-B*K4n);
         temp=1;
         for x=1:max(size(eig1))
            if real(eig1(x))>0
               temp=0;
            elseif real(eig2(x))>0
               temp=0;
            elseif real(eig3(x))>0
               temp=0;
            elseif real(eig4(x))>0
               temp=0;
            end
         end
         if temp==0
            alpha=alpha*0.1;
         elseif temp==1;
            eig1a=eig(A1-B*K1n*Rbar*C);
            eig2a=eig(A2-B*K1n*Rbar*C);
            eig3a=eig(A3-B*K1n*Rbar*C);
            eig4a=eig(A4-B*K1n*Rbar*C);
            temp1=1;
```

```
          for x=1:max(size(eig1))
             if real(eig1a(x))>0
                temp1=0;
             elseif real(eig2a(x))>0
                temp1=0;
             elseif real(eig3a(x))>0
                temp1=0;
             elseif real(eig4a(x))>0
                temp1=0;
             end
          end
          if alpha==1|temp1==1
             state=2;
             fprintf('Initial Kharitonov gain found after %d iterations.\n
             Kinit=K1n;
             Ginit=K1n*Rbar
             transition=1;
          end
          eigstate=1;
       end
    end
    K1=K1n;
    K2=K2n;
    K3=K3n;
    K4=K4n;

% Step 5.  If in state 2, minimize cost function.
  elseif (state==2)&(transition~=1)
    transition=0;
    Jint=trace(P1*Xt)+trace(P2*Xt)+trace(P3*Xt)+trace(P4*Xt);
    J1=0;
    for x=1:N
       J1=J1+r(x,:)*K1*v(:,x)*L(x);
    end
    J2=trace((K2-K1*Cbar)*X2');
    J3=trace((K3-K1*Cbar)*X3');
    J4=trace((K4-K1*Cbar)*X4');
    J(k)=Jint+J1+J2+J3+J4;
    alpha=1;
    Jn=J(k)+1;
    while (Jn>J(k))&(alpha>1e-6)
       K1n=K1+alpha*delK1;
       K2n=K1n*Cbar;
       K3n=K1n*Cbar;
       K4n=K1n*Cbar;
       A1cln=A1-B*K1n;
       A2cln=A2-B*K2n;
       A3cln=A3-B*K3n;
       A4cln=A4-B*K4n;
       P1n=lyap(A1cln',Q+K1n'*R*K1n);
       P2n=lyap(A2cln',Q+K2n'*R*K2n);
       P3n=lyap(A3cln',Q+K3n'*R*K3n);
       P4n=lyap(A4cln',Q+K4n'*R*K4n);
       S1n=lyap(A1cln,Xt);
       S2n=lyap(A2cln,Xt);
```

```
            S3n=lyap(A3cln,Xt);
            S4n=lyap(A4cln,Xt);
            X2n=B'*P2n*S2n-R*K2n*S2n;
            X3n=B'*P3n*S3n-R*K3n*S3n;
            X4n=B'*P4n*S4n-R*K4n*S4n;
            for x=1:N
               Zn(1,x)=r(x,:)*inv(R)*(B'*P1n*S1n+X2n*Cbar'+X3n*Cbar' ...
                  +X4n*Cbar')*inv(S1n)*v(:,x);
               for y=1:N
                  Yn(x,y)=r(y,:)*inv(R)*r(x,:)'*v(:,x)'*inv(S1n)*v(:,y);
               end
            end
            Ln=Zn*inv(Yn);
            Jint=trace(P1n*Xt)+trace(P2n*Xt)+trace(P3n*Xt)+trace(P4n*Xt);
            J1=0;
            for x=1:N
               J1=J1+r(x,:)*K1n*v(:,x)*Ln(x);
            end
            J2=trace((K2n-K1n*Cbar)*X2n');
            J3=trace((K3n-K1n*Cbar)*X3n');
            J4=trace((K4n-K1n*Cbar)*X4n');
            Jn=Jint+J1+J2+J3+J4;
            if Jn>J(k)
               alpha=alpha*0.1;
            end
         end
         if alpha>1e-6
            K1=K1n;
            K2=K2n;
            K3=K3n;
            K4=K4n;
         else
            fprintf('J is minimized within alpha tolerance.\n')
         end
         if k~=0
            Jk=J(k);
         else
            Jn=Jk-1;
         end
      end
      k=k+1;
      if floor(k/100)==(k/100)
         fprintf('%d iterations.  Alpha=%g \n',k,alpha)
      end
   end
   G=K1*Rbar;
   eig1=eig(A1-B*G*C);
   eig2=eig(A2-B*G*C);
   eig3=eig(A3-B*G*C);
   eig4=eig(A4-B*G*C);
   temp=1;
   for x=1:max(size(eig1))
      if real(eig1(x))>0
         temp=0;
      elseif real(eig2(x))>0
```

```
      temp=0;
   elseif real(eig3(x))>0
      temp=0;
   elseif real(eig4(x))>0
      temp=0;
   end
end
fprintf('Total iterations:  %d\n',k-1)
if temp==0;
   fprintf('Algorithm failed -- system not robustly stable.\n')
else
   fprintf('Robustly stable optimal output feedback gain is:\n')
   G
end
```

A.7 charpoly.m

```
% charpoly.m
%
% A "brute force" method to calculate
% the uncertain characteristic polynomial.
%
% Also need the gain matrix:  K
% and the input matrix:  B
%
% Author:  David White

m=0;
firstpass=1;
% Define the uncertain parameters.
q1=[-0.4455,-0.3046];
q2=[0.1003,0.1334];
q3=[-0.02133,-0.01978];
q4=[-0.1179,-0.1092];
q5=[0.01366,0.01473];
q6=[-0.01542,-0.01302];
q7=[0.009628,0.01040];
q8=[0.001663,0.001970];
q9=[-0.1485,-0.1255];
q10=[-0.2917,-0.2794];
q11=[-1.072,-0.9745];
q12=[-0.3033,-0.2757];
% Cycle through each combination to determine the upper
% and lower bounds for uncertain characteristic polynomial.
for a=1:2
 for b=1:2
  for c=1:2
   for d=1:2
    for e=1:2
     for f=1:2
      for g=1:2
       for h=1:2
        for i=1:2
```

```
         for j=1:2
          for k=1:2
           for l=1:2
            m=m+1;
            A=[0 0 0 0 q1(a) 0 0 0 ;
               0 0 0 0 q2(b) q3(c) q4(d) 0 ;
               0 0 0 0 0 q5(e) 0 q6(f) ;
               0 0 0 0 0 0 q7(g) q8(h) ;
               0 0 0 0 q9(i) 0 0 0 ;
               0 0 0 0 0 q10(j) 0 0 ;
               0 0 0 0 0 0 q11(k) 0 ;
               0 0 0 0 0 0 0 q12(l)];
            Acl=A-B*K;
            poles=eig(Acl);
            charpoly=poly(Acl);
            if firstpass
             up=charpoly;
             lo=charpoly;
            else
             for n=1:max(size(charpoly))
              if charpoly(n)>up(n)
               up(n)=charpoly(n);
              elseif charpoly(n)<lo(n)
               lo(n)=charpoly(n);
              end
             end
            end
            firstpass=0;
           end
          end
         end
        end
       end
      end
     end
    end
   end
  end
 end
end
```

A.8 amatrix.m

```
% amatrix.m
%
% A "brute force" method to calculate
% the Kharitonov A matrices.
%
% Author:  David White

m=0;
firstpass=1;
% Define the uncertain parameters.
q1=[-0.4455,-0.3046];
```

```
q2=[0.1003,0.1334];
q3=[-0.02133,-0.01978];
q4=[-0.1179,-0.1092];
q5=[0.01366,0.01473];
q6=[-0.01542,-0.01302];
q7=[0.009628,0.01040];
q8=[0.001663,0.001970];
q9=[-0.1485,-0.1255];
q10=[-0.2917,-0.2794];
q11=[-1.072,-0.9745];
q12=[-0.3033,-0.2757];
% Cycle through each combination to
% determine the upper and lower bounds
% for uncertain characteristic polynomial.
for a=1:2
 for b=1:2
  for c=1:2
   for d=1:2
    for e=1:2
     for f=1:2
      for g=1:2
       for h=1:2
        for i=1:2
         for j=1:2
          for k=1:2
           for l=1:2
            m=m+1;
            A=[0 0 0 0 q1(a) 0 0 0 ;
               0 0 0 0 q2(b) q3(c) q4(d) 0 ;
               0 0 0 0 0 q5(e) 0 q6(f) ;
               0 0 0 0 0 0 q7(g) q8(h) ;
               0 0 0 0 q9(i) 0 0 0 ;
               0 0 0 0 0 q10(j) 0 0 ;
               0 0 0 0 0 0 q11(k) 0 ;
               0 0 0 0 0 0 0 q12(l)];
            poles=eig(A);
            charpoly=poly(A);
            if firstpass
             up=charpoly;
             lo=charpoly;
            else
             for n=1:max(size(charpoly))
              if charpoly(n)>up(n)
               up(n)=charpoly(n);
              elseif charpoly(n)<lo(n)
               lo(n)=charpoly(n);
              end
             end
            end
            firstpass=0;
           end
          end
         end
        end
       end
```

```
      end
     end
    end
   end
  end
 end
end
p=[up;lo];
order=max(size(p));             % order of polynomial
p=fliplr(p);                    % ascending powers of s
K1=[2 2 1 1];
K2=[1 1 2 2];
K3=[1 2 2 1];
K4=[2 1 1 2];
for k=1:order
   m=rem(k,4);
   if m==0
      m=4;
   end
   p1(k)=p(K1(m),k);            % Generate Kharitonov
   p2(k)=p(K2(m),k);            % polynomials by using
   p3(k)=p(K3(m),k);            % the appropriate rows
   p4(k)=p(K4(m),k);            % of p.
end
% Generate Kharitonov matrices.
A1=[zeros(order-2,1) eye(order-2);-p1(1:order-1)/p1(order)];
A2=[zeros(order-2,1) eye(order-2);-p2(1:order-1)/p2(order)];
A3=[zeros(order-2,1) eye(order-2);-p3(1:order-1)/p3(order)];
A4=[zeros(order-2,1) eye(order-2);-p4(1:order-1)/p4(order)];

% And if you want it, calculate the upper and
% lower bounds for the uncertain A matrix.
up=fliplr(up);
lo=fliplr(lo);
Aup=[zeros(order-2,1) eye(order-2);-up(1:order-1)/up(order)];
Alo=[zeros(order-2,1) eye(order-2);-lo(1:order-1)/lo(order)];
```

Appendix B

Simulation Results

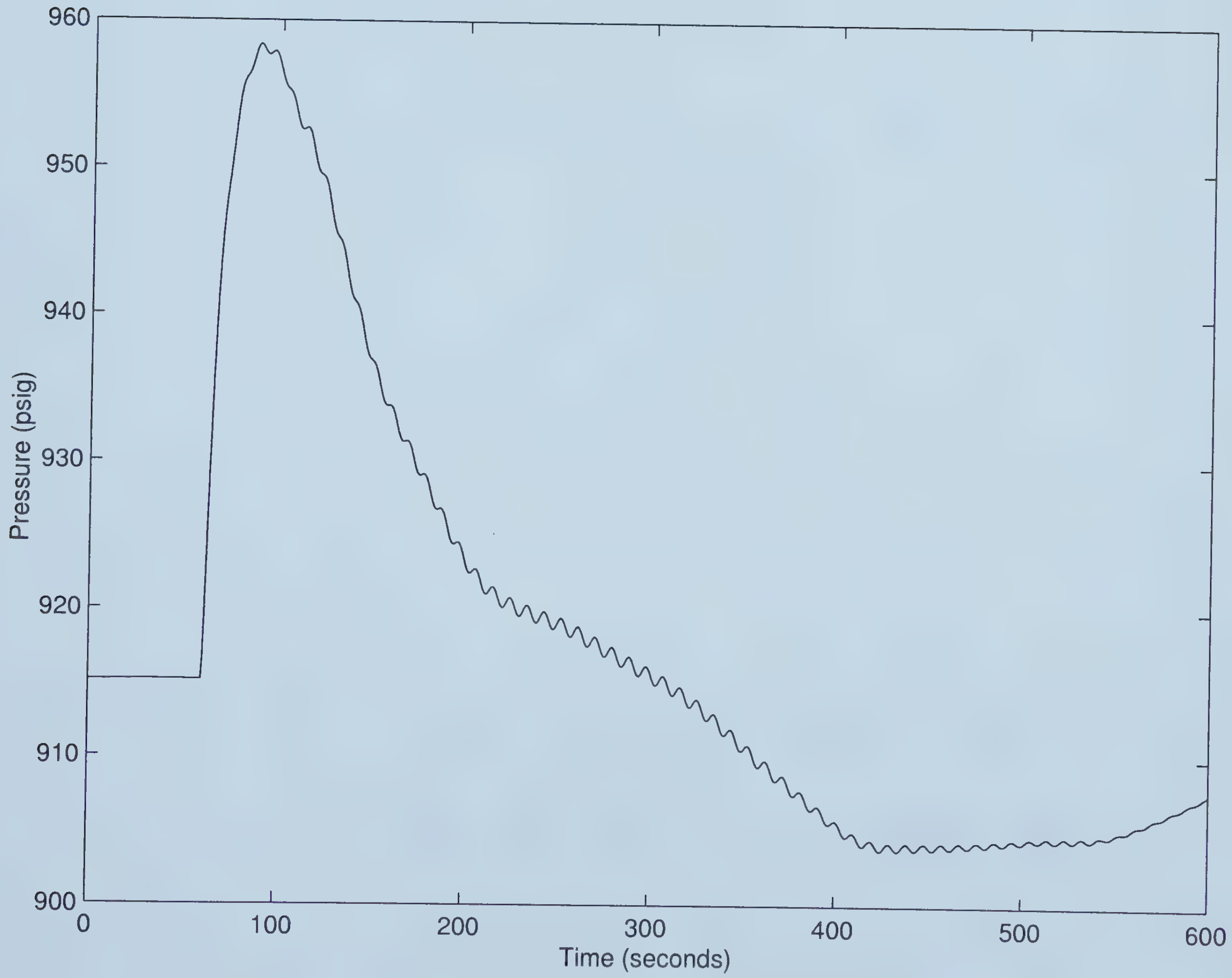

Figure B.1: 900# header pressure response with the existing controllers.

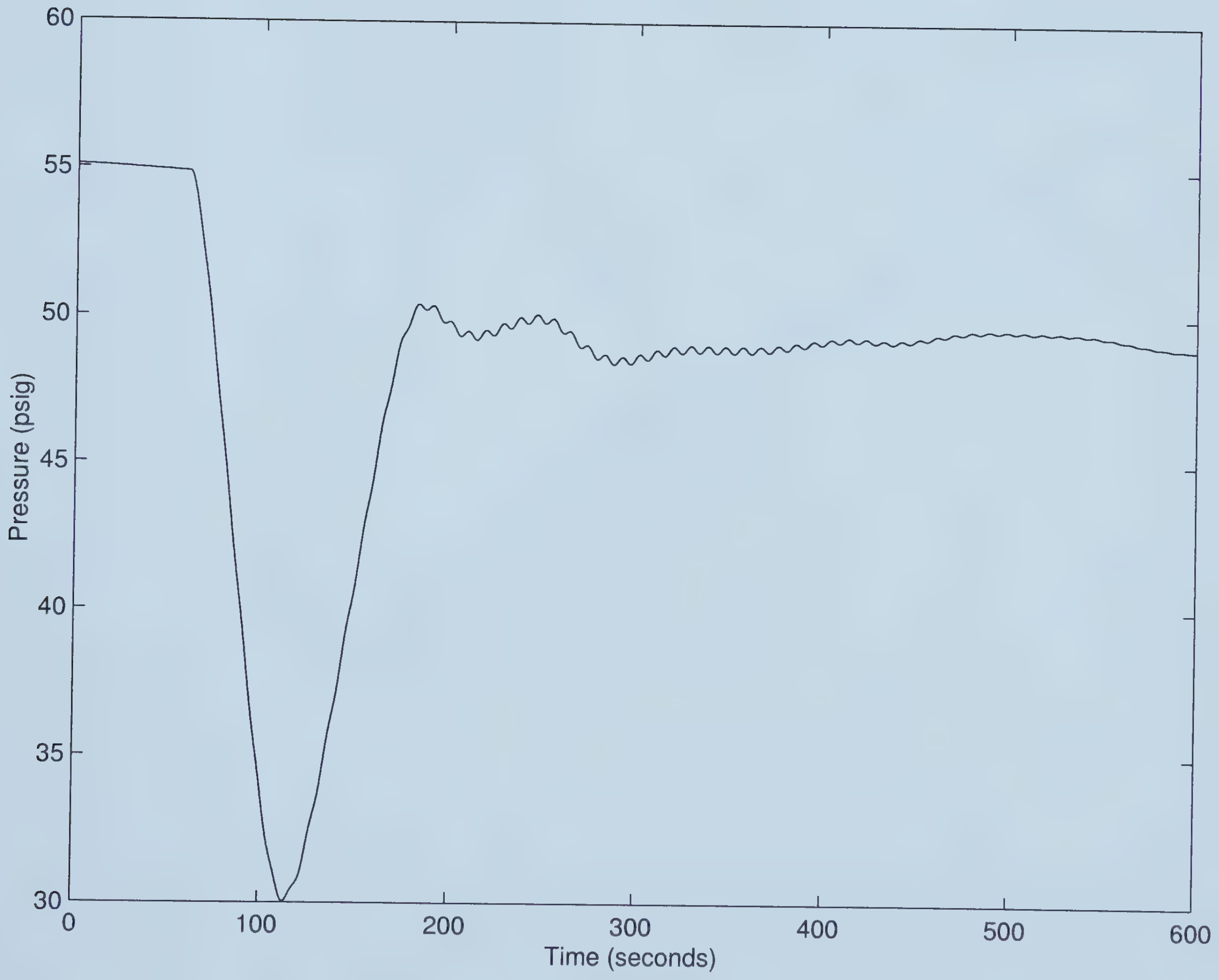

Figure B.2: 50# header pressure response with the existing controllers.

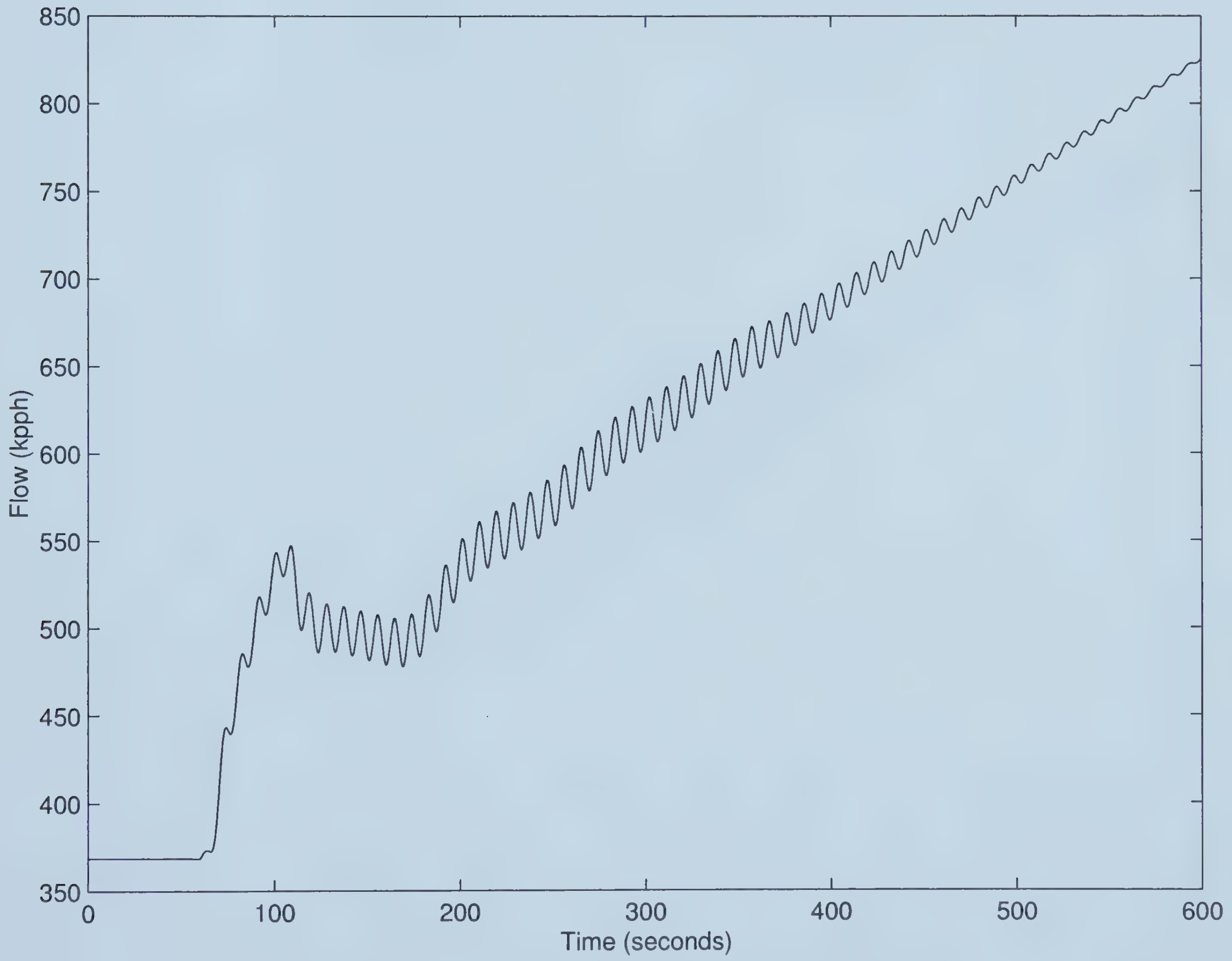

Figure B.3: 900/600# flow response with the existing controllers.

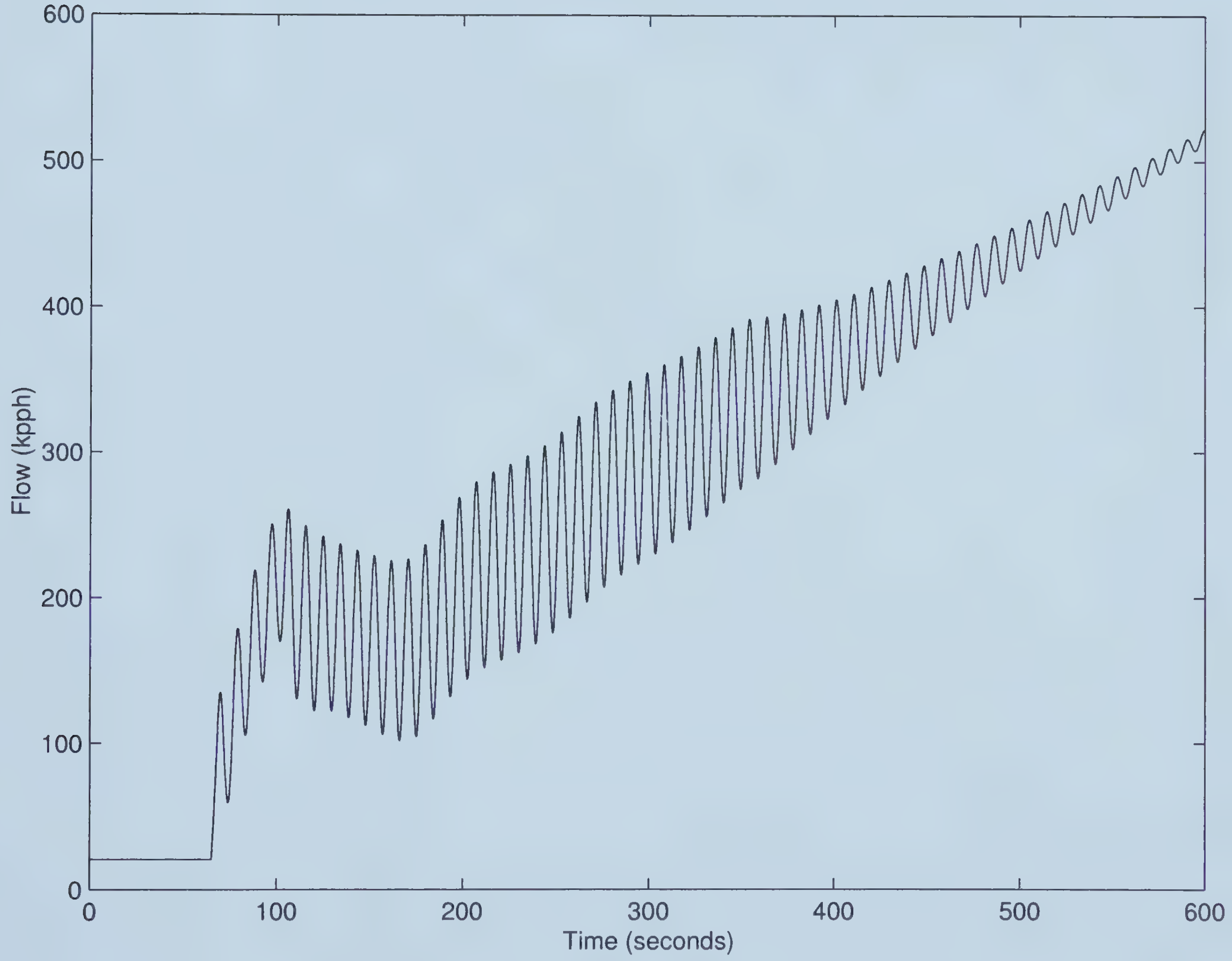

Figure B.4: 600/50# flow response with the existing controllers.

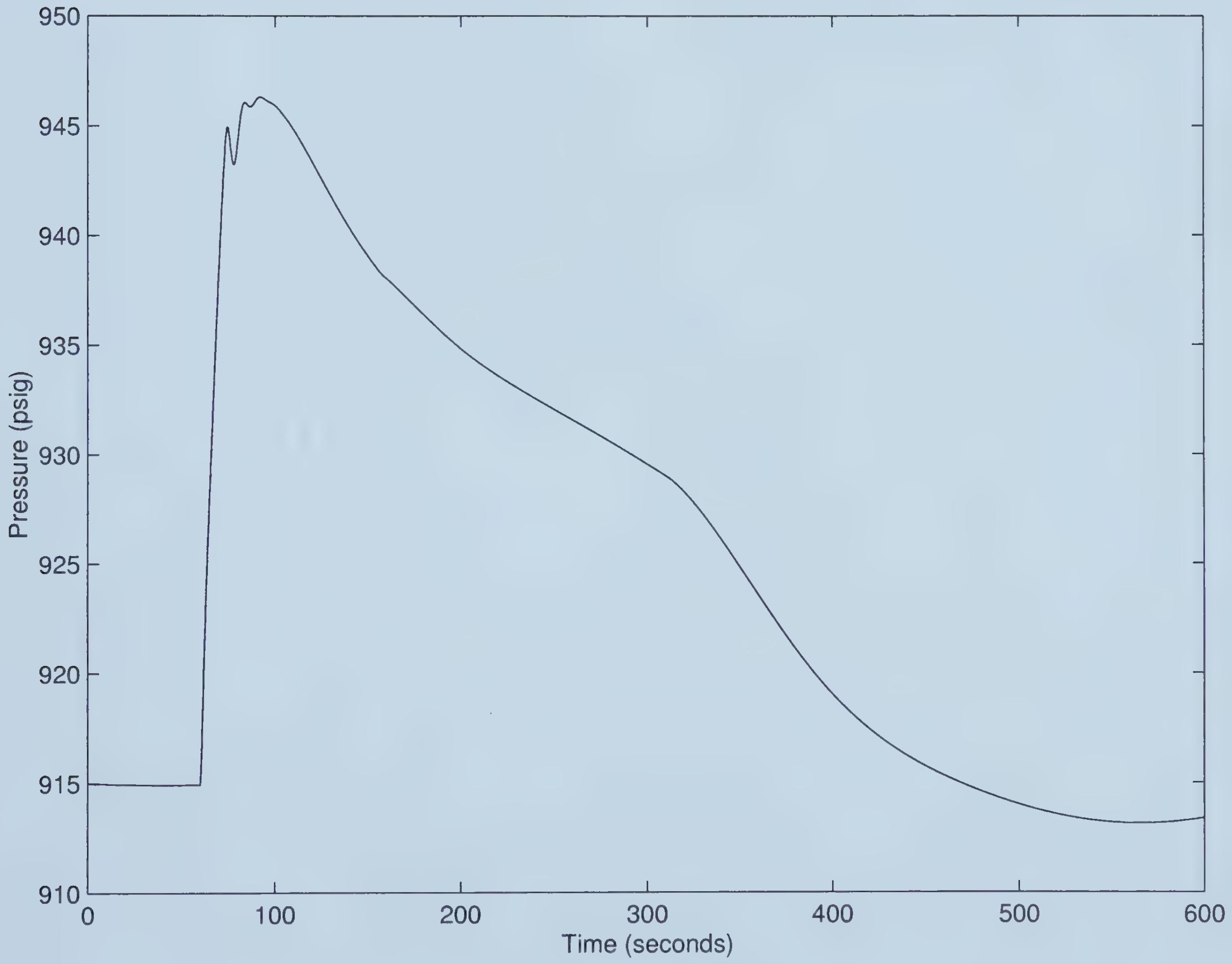

Figure B.5: 900# header pressure response with the state feedback controller.

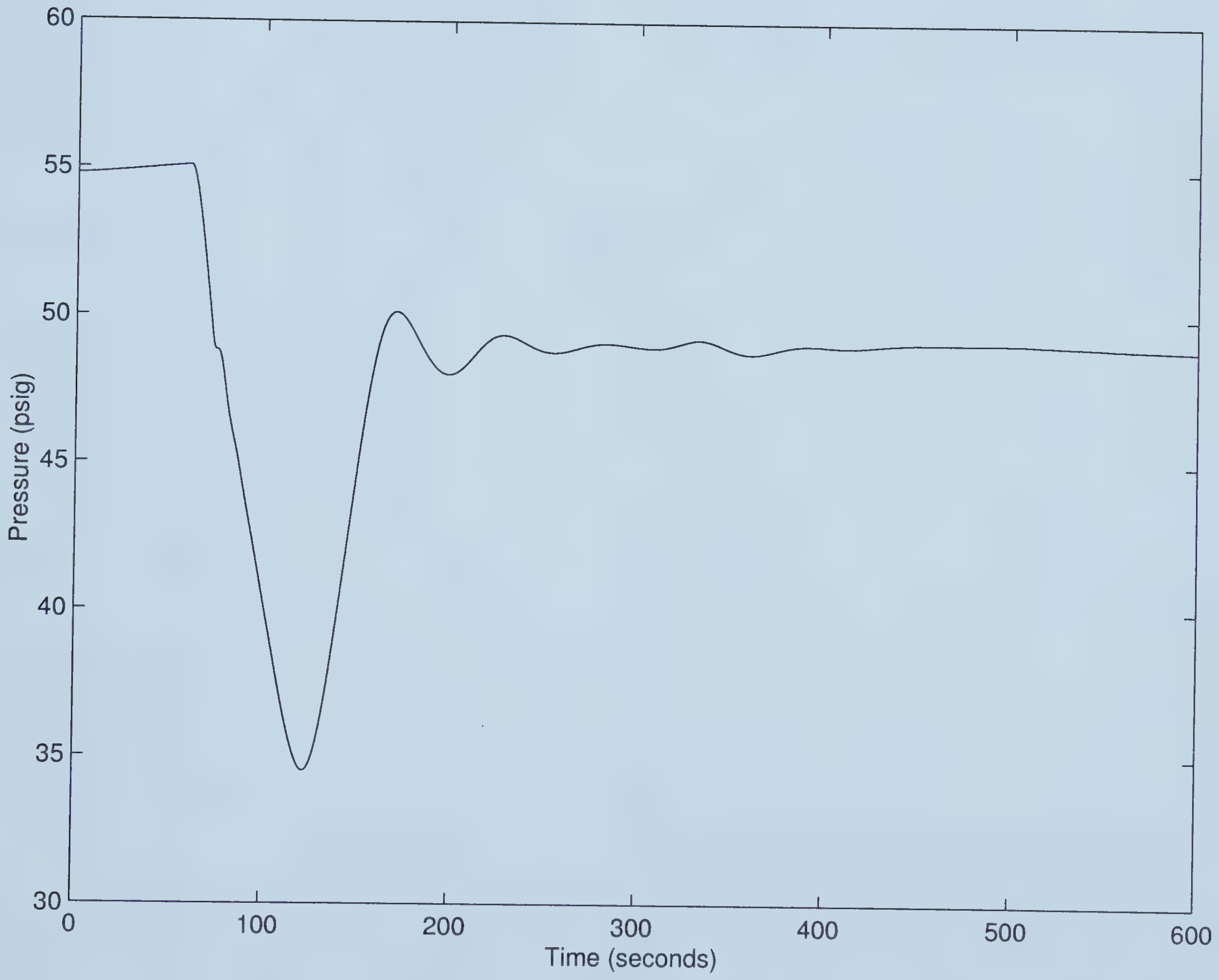

Figure B.6: 50# header pressure response with the state feedback controller.

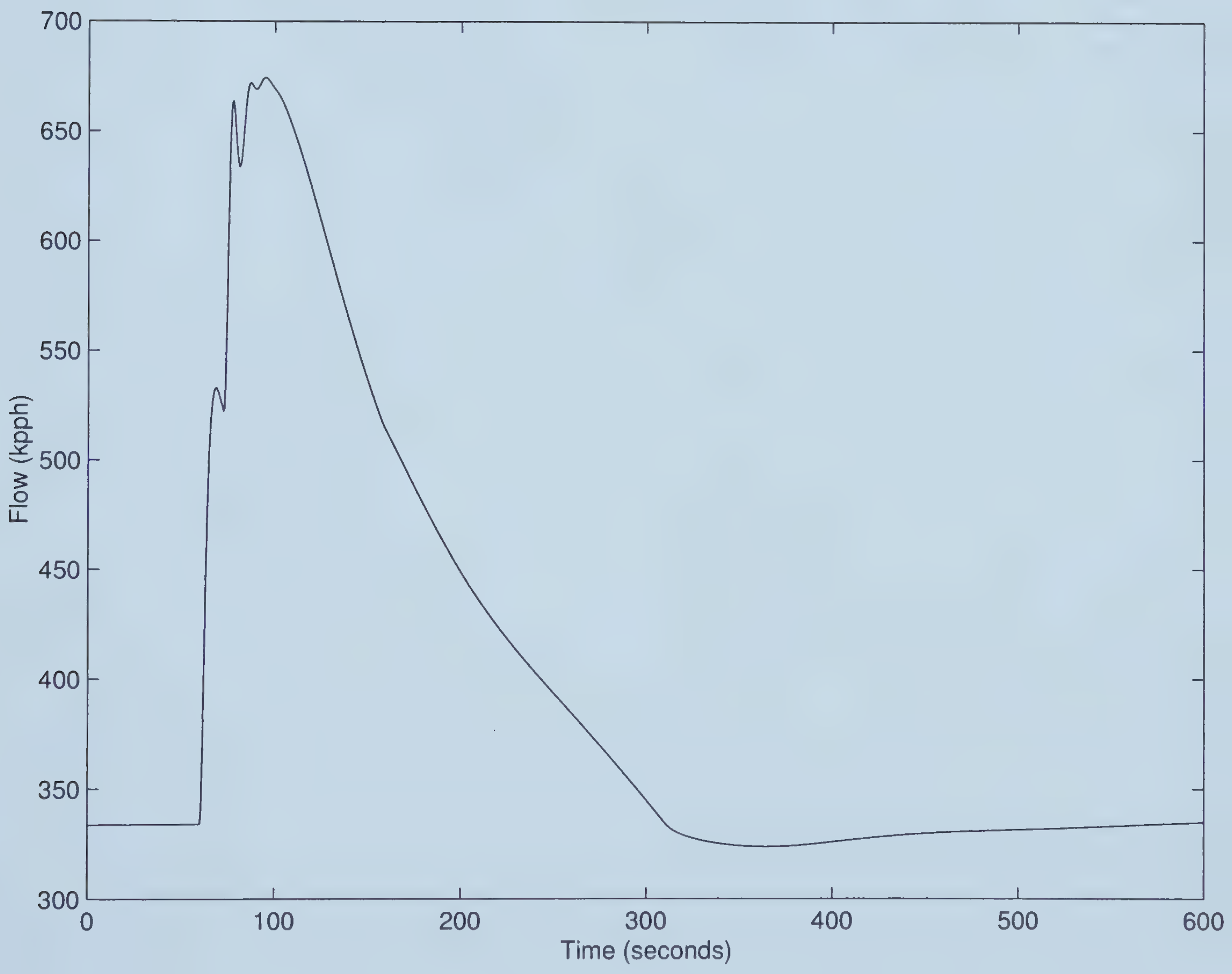

Figure B.7: 900/600# flow response with the state feedback controller.

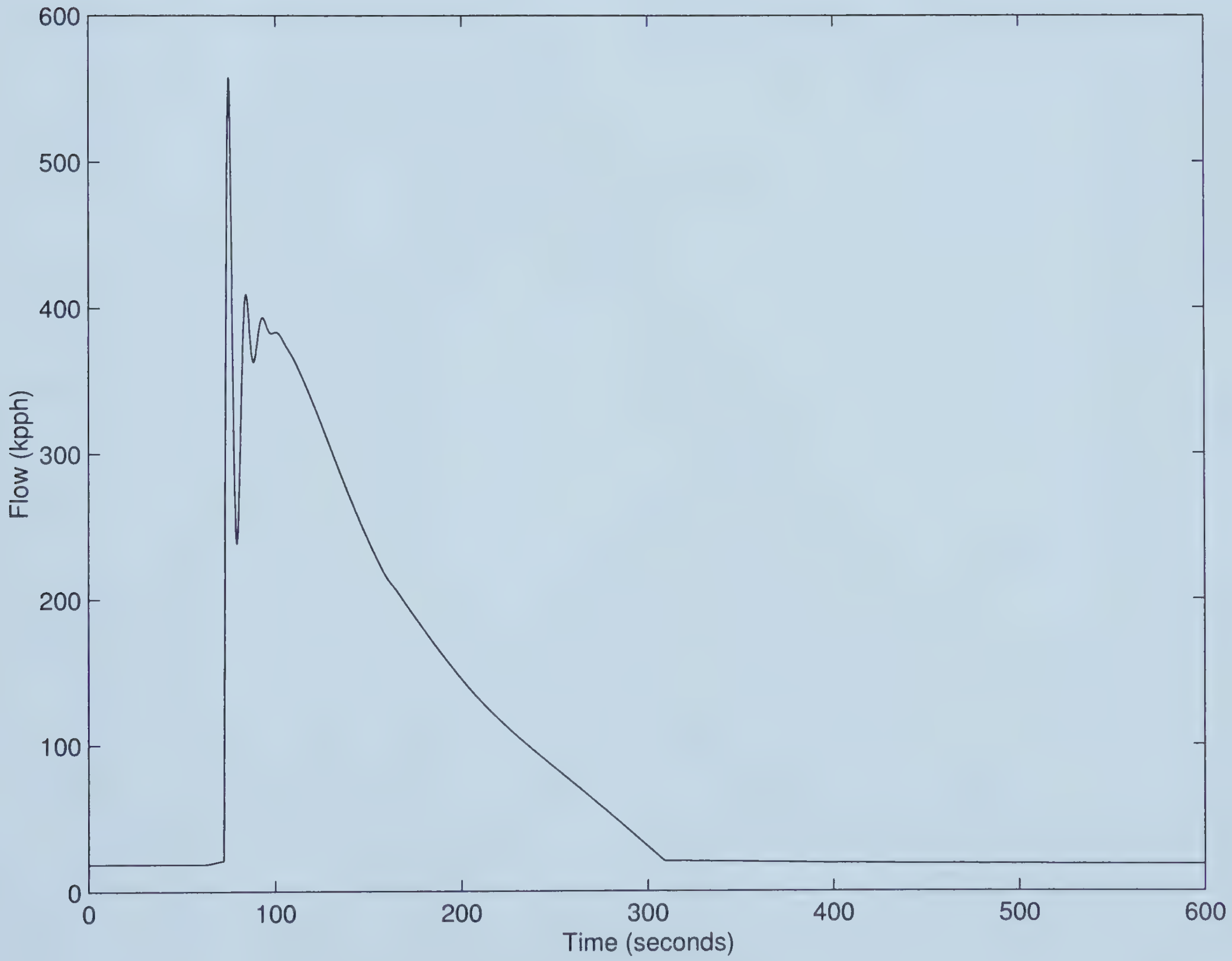

Figure B.8: 600/50# flow response with the state feedback controller.

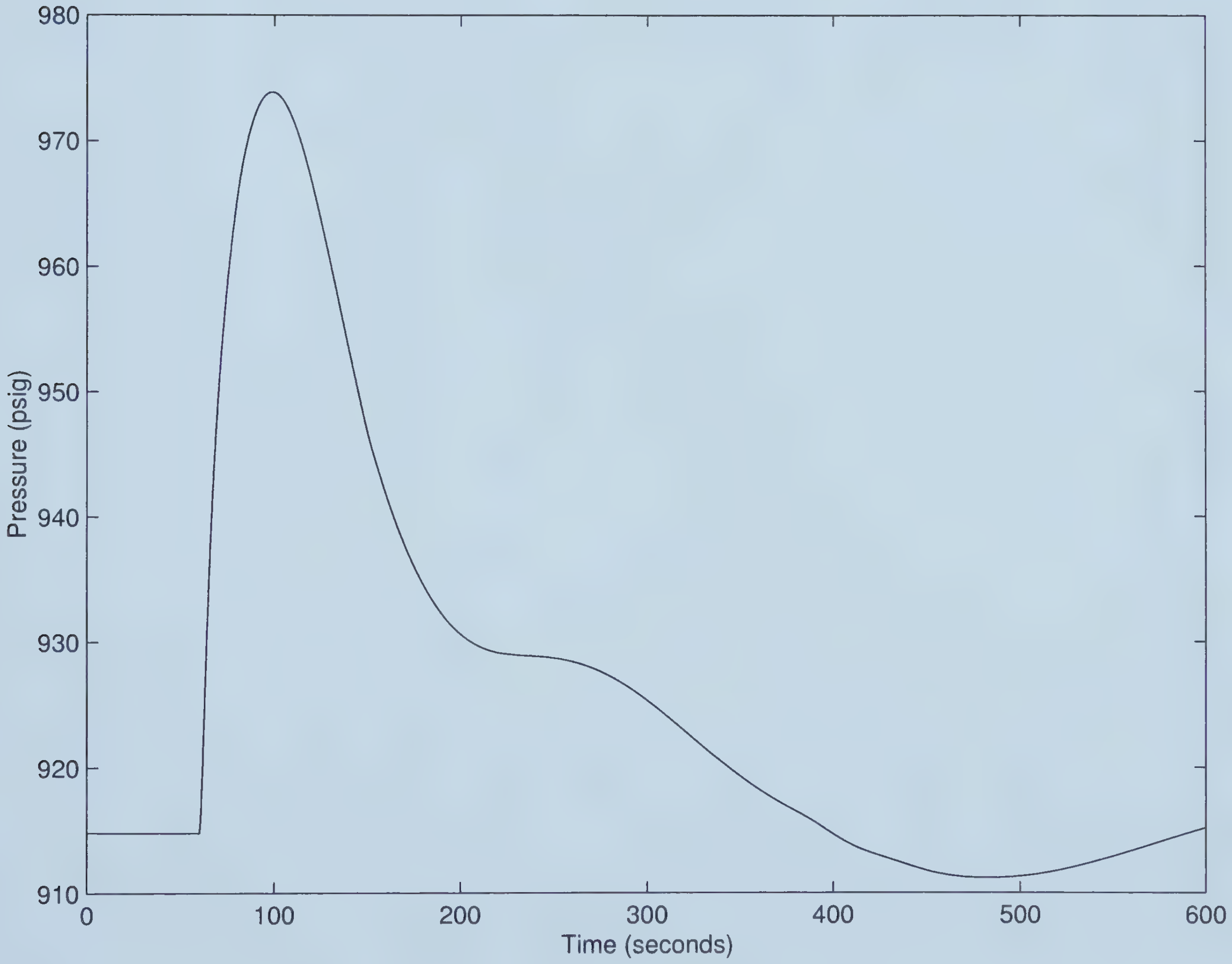

Figure B.9: 900# header pressure response with the output feedback controller.

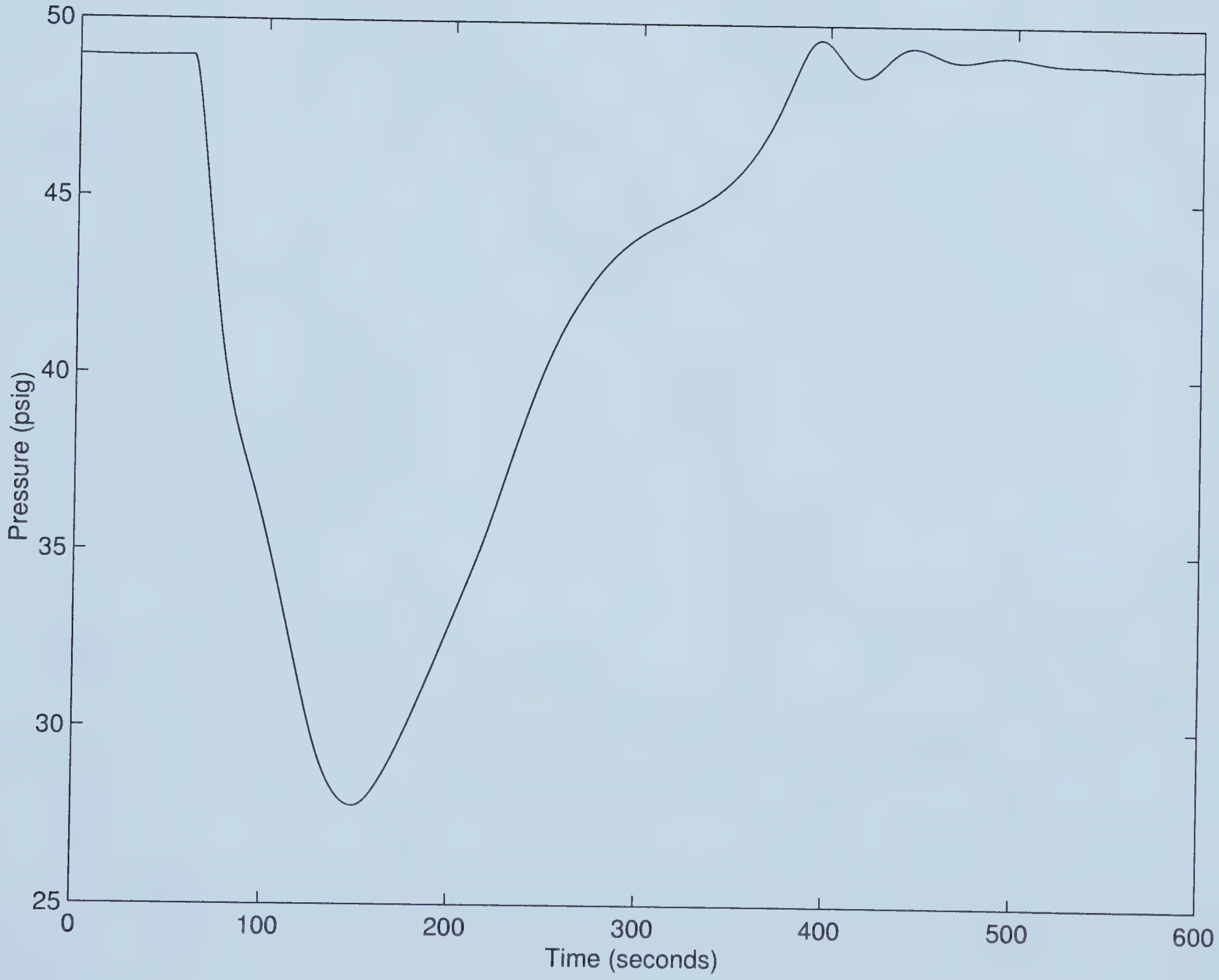

Figure B.10: 50# header pressure response with the output feedback controller.

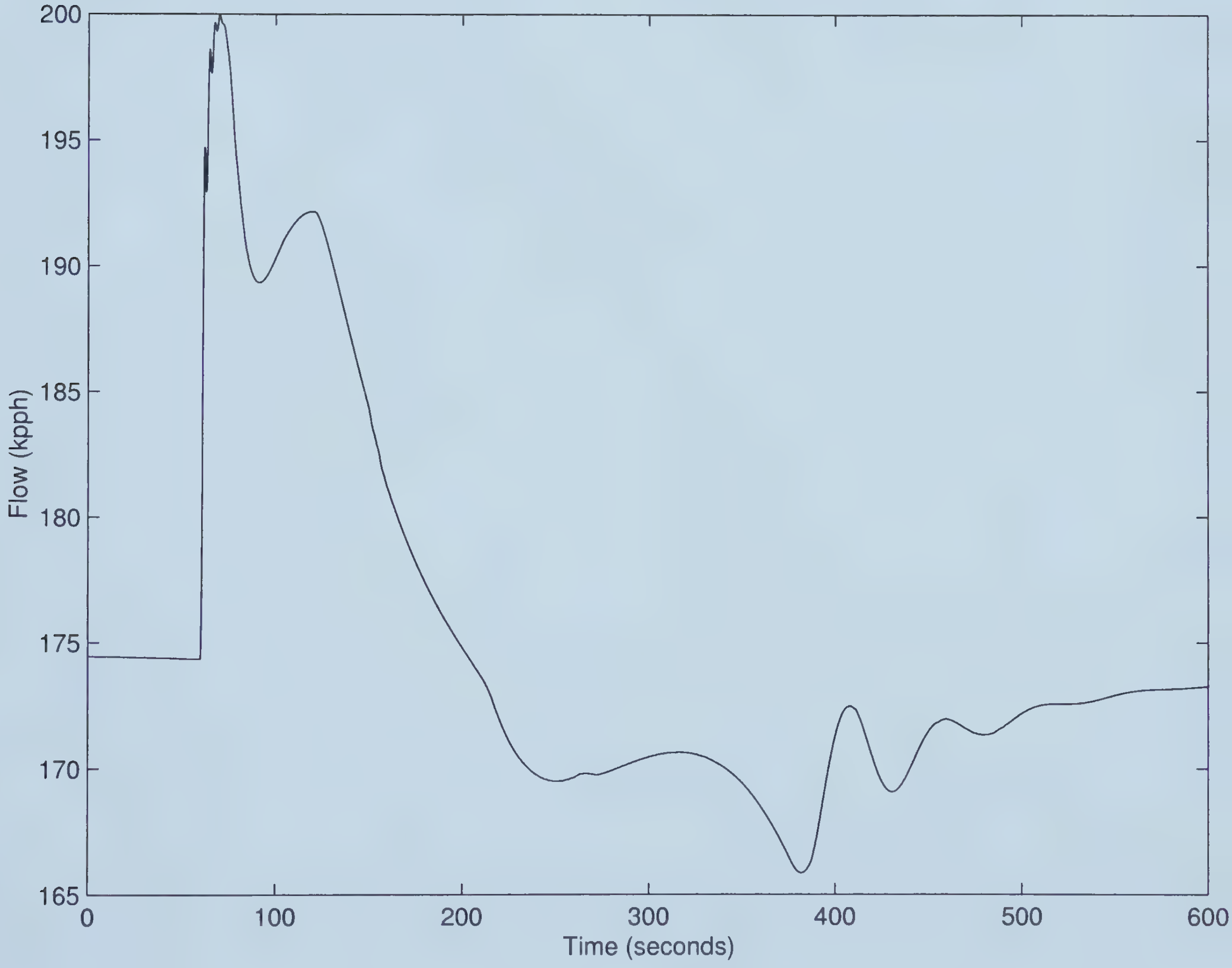

Figure B.11: 900/600# flow response with the output feedback controller.

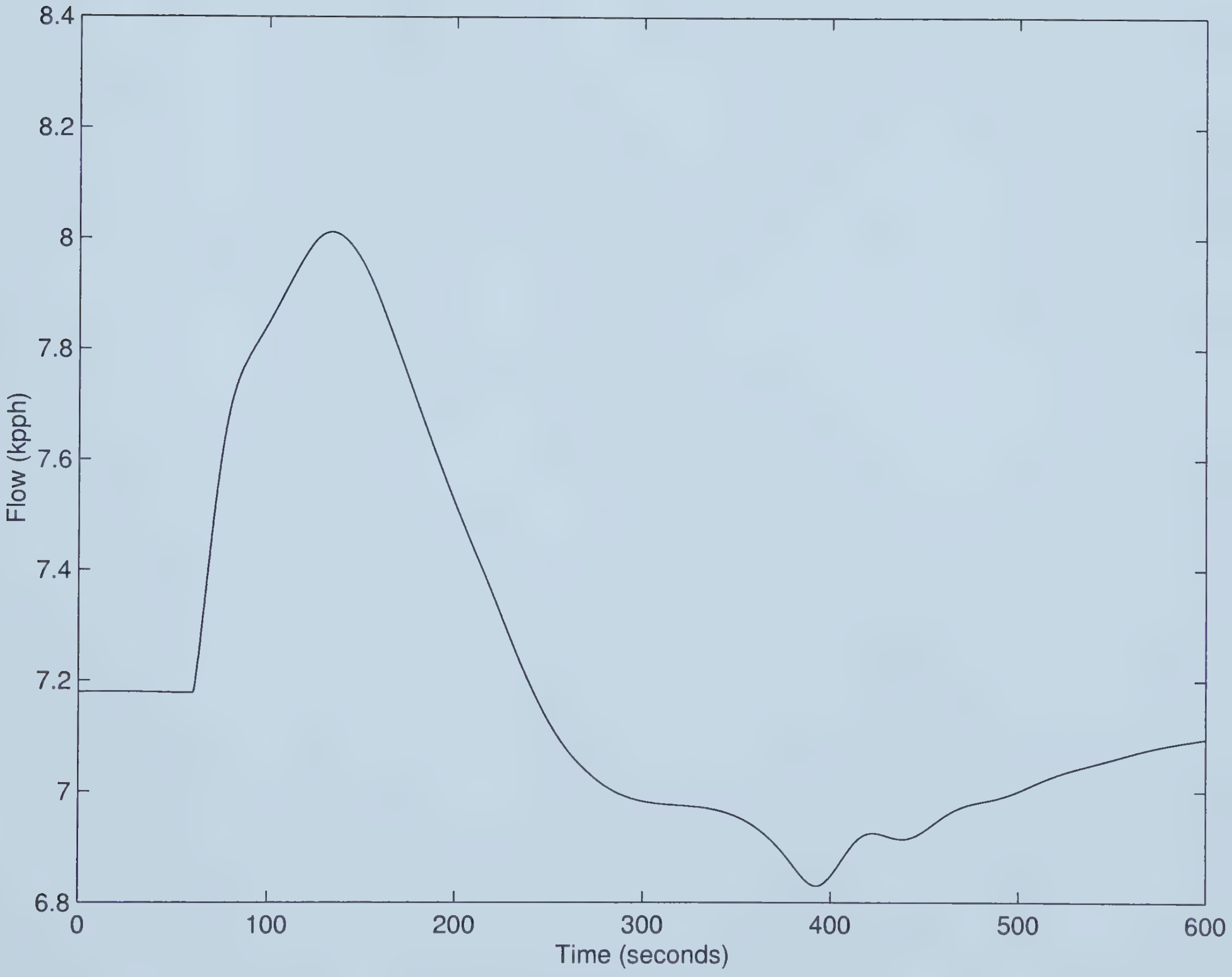

Figure B.12: 600/50# flow response with the output feedback controller.